```
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBB          BBB   AAA       AAA   SSS           RRR         RRR       TTT        LLL
BBB          BBB   AAA       AAA   SSS           RRR         RRR       TTT        LLL
BBB          BBB   AAA       AAA   SSS           RRR         RRR       TTT        LLL
BBB          BBB   AAA       AAA   SSS           RRR         RRR       TTT        LLL
BBB          BBB   AAA       AAA   SSS           RRR         RRR       TTT        LLL
BBB          BBB   AAA       AAA   SSS           RRR         RRR       TTT        LLL
BBBBBBBBBBBB       AAA       AAA     SSSSSSSSS   RRRRRRRRRRR           TTT        LLL
BBBBBBBBBBBB       AAA       AAA     SSSSSSSSS   RRRRRRRRRRR           TTT        LLL
BBBBBBBBBBBB       AAA       AAA     SSSSSSSSS   RRRRRRRRRRR           TTT        LLL
BBB          BBB   AAAAAAAAAAAAAAA           SSS  RRR   RRR             TTT        LLL
BBB          BBB   AAAAAAAAAAAAAAA           SSS  RRR   RRR             TTT        LLL
BBB          BBB   AAAAAAAAAAAAAAA           SSS  RRR   RRR             TTT        LLL
BBB          BBB   AAA       AAA           SSS  RRR      RRR           TTT        LLL
BBB          BBB   AAAAAAAAAAAAAAA           SSS  RRR   RRR             TTT        LLL
BBB          BBB   AAA       AAA           SSS  RRR      RRR           TTT        LLL
BBB          BBB   AAA       AAA           SSS  RRR      RRR           TTT        LLL
BBB          BBB   AAA       AAA           SSS  RRR      RRR           TTT        LLL
BBBBBBBBBBBB       AAA       AAA   SSSSSSSSSSS   RRR         RRR       TTT        LLLLLLLLLLLLLLL
BBBBBBBBBBBB       AAA       AAA   SSSSSSSSSSS   RRR         RRR       TTT        LLLLLLLLLLLLLLL
BBBBBBBBBBBB       AAA       AAA   SSSSSSSSSSS   RRR         RRR       TTT        LLLLLLLLLLLLLLL
```

```
BBBBBBBB      AAAAAA      SSSSSSSS  VV      VV    IIIIII    RRRRRRRR    TTTTTTTTTT  UU      UU    AAAAAA
BBBBBBBB      AAAAAA      SSSSSSSS  VV      VV    IIIIII    RRRRRRRR    TTTTTTTTTT  UU      UU    AAAAAA
BB      BB   AA      AA   SS        VV      VV      II      RR      RR      TT      UU      UU   AA      AA
BB      BB   AA      AA   SS        VV      VV      II      RR      RR      TT      UU      UU   AA      AA
BB      BB   AA      AA   SS        VV      VV      II      RR      RR      TT      UU      UU   AA      AA
BB      BB   AA      AA   SS        VV      VV      II      RR      RR      TT      UU      UU   AA      AA
BBBBBBBB     AA      AA     SSSSSS   VV      VV      II      RRRRRRRR        TT      UU      UU   AA      AA
BBBBBBBB     AA      AA     SSSSSS   VV      VV      II      RRRRRRRR        TT      UU      UU   AA      AA
BB      BB   AAAAAAAAAA           SS  VV      VV      II      RR   RR         TT      UU      UU   AAAAAAAAAA
BB      BB   AAAAAAAAAA           SS  VV      VV      II      RR   RR         TT      UU      UU   AAAAAAAAAA
BB      BB   AA      AA           SS    VV  VV        II      RR    RR        TT      UU      UU   AA      AA
BB      BB   AA      AA           SS    VV  VV        II      RR    RR        TT      UU      UU   AA      AA
BBBBBBBB     AA      AA   SSSSSSSS      VV          IIIIII    RR      RR      TT      UUUUUUUUUU   AA      AA
BBBBBBBB     AA      AA   SSSSSSSS      VV          IIIIII    RR      RR      TT      UUUUUUUUUU   AA      AA

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

```
;     1          0001  0 MODULE BAS$VIRTUAL_ARR (                      ! Virtual Array interface
;     2          0002  0                 IDENT = '1-033'              ! File: BASVIRTUA.B32 Edit: DG1033
;     3          0003  0                 ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !*****************************************************************************
;     7          0007  1 !*                                                                           *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                      *
;    11          0011  1 !*                                                                           *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;    17          0017  1 !*  TRANSFERRED.                                                              *
;    18          0018  1 !*                                                                           *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    21          0021  1 !*  CORPORATION.                                                              *
;    22          0022  1 !*                                                                           *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*                                                                           *
;    27          0027  1 !*****************************************************************************
;    28          0028  1 !
;    29          0029  1
;    30          0030  1 !++
;    31          0031  1 ! FACILITY:  VAX-11 BASIC Virtual Array Support
;    32          0032  1 !
;    33          0033  1 ! ABSTRACT:
;    34          0034  1 !
;    35          0035  1 !       This module contains the support for VAX-11 BASIC virtual
;    36          0036  1 !       arrays.  This consists of the various FETCH and STORE routines
;    37          0037  1 !       called by compiled code and the interface to the I/O system.
;    38          0038  1 !
;    39          0039  1 ! ENVIRONMENT:  VAX-11 User Mode
;    40          0040  1 !
;    41          0041  1 ! AUTHOR: John Sauter, CREATION DATE: 02-FEB-1979
;    42          0042  1 !
;    43          0043  1 ! MODIFIED BY:
;    44          0044  1 !
;    45          0045  1 ! 1-001 - Original.  This version handles only non-virtual arrays.
;    46          0046  1 !           JBS 02-FEB-1979
;    47          0047  1 ! 1-002 - Make the JSB entry points take arguments in registers.
;    48          0048  1 !           JBS 26-FEB-1979
;    49          0049  1 ! 1-003 - Take linkage definitions from BASVIRTUA.REQ.  JBS 20-FEB-1979
;    50          0050  1 ! 1-004 - Based on a review of the virtual array concept with part
;    51          0051  1 !           of the VAX ECO board, virtual arrays are a BASIC-only
;    52          0052  1 !           feature.  Therefore, change OTS$ to BAS$ in the entry
;    53          0053  1 !           points.  Also, only longwords need be passed as indexes.
;    54          0054  1 !           JBS 03-APR-1979
;    55          0055  1 ! 1-005 - Since virtual array descriptors will have their indicies
;    56          0056  1 !           computed in the same way as ordinary arrays, rearrange
;    57          0057  1 !           the code to compute the location in the array of the value
```

```
;    58          0058  1 !           before checking the class of the descriptor.  This also
;    59          0059  1 !           helps support arrays of dynamic strings.  JBS 03-APR-1979
;    60          0060  1 ! 1-006 - Add calls to internal virtual array subroutines.  These are
;    61          0061  1 !           just stubs for now.  JBS 04-APR-1979
;    62          0062  1 ! 1-007 - Call temporary I/O routines so we can check out the indexing
;    63          0063  1 !           subroutines.  JBS 04-APR-1979
;    64          0064  1 ! 1-008 - Make those temporary I/O routines permanent, and handle
;    65          0065  1 !           trailing NULs properly when fetching and storing strings.
;    66          0066  1 !           JBS 18-MAY-1979
;    67          0067  1 ! 1-009 - Change OTS$S to STR$ facility.  JBS 18-MAY-1979
;    68          0068  1 ! 1-010 - Change STR$COPY_R_DX to STR$COPY_R  JBS 20-MAY-1979
;    69          0069  1 ! 1-011 - Correct a typo which caused normal array fetches to fail.
;    70          0070  1 !           JBS 31-MAY-1979
;    71          0071  1 ! 1-012 - Correct the local definition of DSC$K_DTYPE_DSC.  We must
;    72          0072  1 !           remember to remove this when the definition goes into
;    73          0073  1 !           STARLE.  JBS 01-JUN-1979
;    74          0074  1 ! 1-013 - Make the index parameters to BAS$FETCH_VA and BAS$STORE_VA
;    75          0075  1 !           be by value.  JBS 01-JUN-1979
;    76          0076  1 ! 1-014 - Correct an error in storing a string.  JBS 05-JUN-1979
;    77          0077  1 ! 1-015 - Debug string virtual arrays.  JBS 11-JUN-1979
;    78          0078  1 ! 1-016 - Remove local definitions of DSC$ symbols.  JBS 19-JUN-1979
;    79          0079  1 ! 1-017 - Use BASLNK.  JBS 26-JUN-1979
;    80          0080  1 ! 1-018 - Add BAS$STO_FA_RDX.  JBS 13-JUL-1979
;    81          0081  1 ! 1-019 - Change calls to STR$COPY.  JBS 19-JUL-1979
;    82          0082  1 ! 1-020 - Change BAS$$COPY_D and BAS$$COPY_F to end in _R1.  JBS 23-AUG-1979
;    83          0083  1 ! 1-021 - Allow the INDEX2 argument of BAS$STO_FA_RDX to be zero.
;    84          0084  1 !           JBS 06-NOV-1979
;    85          0085  1 ! 1-022 - Speed up storing into a string virtual array by creating the
;    86          0086  1 !           nulls with a single call to STR$DUPL_CHAR instead of with
;    87          0087  1 !           multiple calls to STR$CONCAT.  JBS 19-MAY-1980
;    88          0088  1 ! 1-023 - Add support for byte, g and h floating.  PLL 9-Sep-81
;    89          0089  1 ! 1-024 - Add support for records.  Dtype Z indicates a record -
;    90          0090  1 !           it is handled the same as text.  PLL 26-Feb-82
;    91          0091  1 ! 1-025 - Fix bug from edit 024 - Dtype Z required extending the range
;    92          0092  1 !           of CASE statements.  PLL 5-Mar-82
;    93          0093  1 ! 1-026 - Add support for decimal arrays.  Decimal lengths are the number
;    94          0094  1 !           of digits (not including the sign) rather than the number of bytes,
;    95          0095  1 !           so the calculation of the linear index is slightly different.
;    96          0096  1 !           PLL 12-Mar-82
;    97          0097  1 ! 1-027 - Remove a couple of dots from the ASHP instruction.  PLL 12-Apr-82
;    98          0098  1 ! 1-028 - The last edit corrected FETCH_BFA - also correct STORE_BFA.
;    99          0099  1 !           12-Apr-82
;   100          0100  1 ! 1-029 - Change call to VA_FETCH, VA_STORE for decimal arrays.  PLL 12-Apr-82
;   101          0101  1 ! 1-030 - Add entry point BAS$STORE_BFA_OFF.  PLL 8-Jun-1982
;   102          0102  1 ! 1-031 - Modify BAS$STORE_BFA_OFF so that no checking is performed on
;   103          0103  1 !           the source and destination data types.  PLL 26-Jul-1982
;   104          0104  1 ! 1-032 - modified all length calculations that used to round up to the
;   105          0105  1 !           nearest multple of 2 to round up to the nearest POWER of 2.
;   106          0106  1 !           added a kludge to dummy up the length of a record in the array
;   107          0107  1 !           descr just prior to the call to BAS$VA_STORE in BAS$STORE_BFA_OFF.
;   108          0108  1 !           MDL 3-Aug-1982
;   109          0109  1 ! 1-033 - Add support for dynamically mapped arrays.  DG 6-Feb-1984
;   110          0110  1 !           NOTE - these changes did not have to be incorporated into the
;   111          0111  1 !           BAS$STO_FA... routines because the routine that calls the STORE
;   112          0112  1 !           routines specifically calls BAS$STORE_BFA for dynamically mapped
;   113          0113  1 !           arrays.
;   114          0114  1 !--
```

```
;  115          0115  1
;  116          0116  1 !<BLF/PAGE>
```

```
;  118          0117  1 !
;  119          0118  1 ! SWITCHES:
;  120          0119  1 !
;  121          0120  1
;  122          0121  1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
;  123          0122  1
;  124          0123  1 !
;  125          0124  1 ! LINKAGES:
;  126          0125  1 !
;  127          0126  1
;  128          0127  1 LINKAGE
;  129          0128  1     COPY_JSB = JSB (REGISTER = 0, REGISTER = 1) !
;  130          0129  1     : NOTUSED (2, 3, 4, 5, 6, 7, 8, 9, 10, 11);
;  131          0130  1
;  132          0131  1 REQUIRE 'RTLIN:BASLNK';                         ! Define linkage VA_JSB
;  133          0208  1
;  134          0209  1 !
;  135          0210  1 ! TABLE OF CONTENTS:
;  136          0211  1 !
;  137          0212  1
;  138          0213  1 FORWARD ROUTINE
;  139          0214  1     BAS$FETCH_BFA : NOVALUE,                    ! Fetch a value by descriptor
;  140          0215  1     BAS$STORE_BFA : NOVALUE,                    ! Store a value by descriptor
;  141          0216  1     BAS$STORE_BFA_OFF : NOVALUE,                ! Store value by desc w/offset
;  142          0217  1     BAS$STO_FA_RDX : NOVALUE,                   ! Store a string by reference
;  143          0218  1     BAS$FET_FA_W_R8 : VA_JSB,                   ! Fetch a word
;  144          0219  1     BAS$STO_FA_W_R8 : VA_JSB NOVALUE,           ! Store a word
;  145          0220  1     BAS$FET_FA_L_R8 : VA_JSB,                   ! Fetch a longword
;  146          0221  1     BAS$STO_FA_L_R8 : VA_JSB NOVALUE,           ! Store a longword
;  147          0222  1     BAS$FET_FA_F_R8 : VA_JSB,                   ! Fetch a floating value
;  148          0223  1     BAS$STO_FA_F_R8 : VA_JSB NOVALUE,           ! Store a floating value
;  149          0224  1     BAS$FET_FA_D_R8 : VA_JSB NOVALUE,           ! Fetch a double-floating value
;  150          0225  1     BAS$STO_FA_D_R8 : VA_JSB NOVALUE,           ! Store a double-floating value
;  151          0226  1     BAS$FET_FA_B_R8 : VA_JSB,                   ! Fetch a byte
;  152          0227  1     BAS$STO_FA_B_R8 : VA_JSB NOVALUE,           ! Store a byte
;  153          0228  1     BAS$FET_FA_G_R8 : VA_JSB NOVALUE,           ! Fetch a g floating value
;  154          0229  1     BAS$STO_FA_G_R8 : VA_JSB NOVALUE,           ! Store a g floating value
;  155          0230  1     BAS$FET_FA_H_R8 : VA_JSB NOVALUE,           ! Fetch an h floating value
;  156          0231  1     BAS$STO_FA_H_R8 : VA_JSB NOVALUE;           ! Store an h floating value
;  157          0232  1
;  158          0233  1 !
;  159          0234  1 ! INCLUDE FILES:
;  160          0235  1 !
;  161          0236  1
;  162          0237  1 REQUIRE 'RTLIN:RTLPSECT';                       ! Macros for defining psects
;  163          0332  1
;  164          0333  1 LIBRARY 'RTLSTARLE';                            ! System symbols
;  165          0334  1
;  166          0335  1 !
;  167          0336  1 ! MACROS:
;  168          0337  1 !
;  169          0338  1 !       NONE
;  170          0339  1 !
;  171          0340  1 ! EQUATED SYMBOLS:
;  172          0341  1 !
;  173          0342  1 !       NONE
;  174          0343  1 !
```

```
;  175          0344  1 ! PSECTS:
;  176          0345  1 !
;  177          0346  1 DECLARE_PSECTS (BAS);                   ! Declare psects for BAS$ facility
;  178          0347  1 !
;  179          0348  1 ! OWN STORAGE:
;  180          0349  1 !
;  181          0350  1 !       NONE
;  182          0351  1 !
;  183          0352  1 ! EXTERNAL REFERENCES:
;  184          0353  1 !
;  185          0354  1
;  186          0355  1 EXTERNAL ROUTINE
;  187          0356  1     BAS$$STOP : NOVALUE,                ! signals fatal error
;  188          0357  1     BAS$$COPY_F_R1 : COPY_JSB NOVALUE,  ! Copy a floating number
;  189          0358  1     BAS$$COPY_D_R1 : COPY_JSB NOVALUE,  ! Copy a double number
;  190          0359  1     BAS$$COPY_G_R1 : COPY_JSB NOVALUE,  ! Copy a g float number
;  191          0360  1     BAS$$COPY_H_R3 : COPY_JSB NOVALUE,  ! Copy an h float number
;  192          0361  1     BAS$$VA_FETCH : NOVALUE,            ! Fetch from virt. array
;  193          0362  1     BAS$$VA_STORE : NOVALUE,            ! Store in virt. array
;  194          0363  1     STR$GETT_DX,                        ! Allocate a string
;  195          0364  1     STR$FREET_DX,                       ! Deallocate a string
;  196          0365  1     STR$COPY_DX,                        ! Copy by descriptor
;  197          0366  1     STR$COPY_R,                         ! Copy by reference
;  198          0367  1     STR$CONCAT,                         ! Concatenate two strings
;  199          0368  1     STR$DUPL_CHAR;                      ! Make lots of a character
;  200          0369  1
;  201          0370  1 !+
;  202          0371  1 ! The following are the error codes used in this module.
;  203          0372  1 !-
;  204          0373  1
;  205          0374  1 EXTERNAL LITERAL
;  206          0375  1     BAS$K_MATARRTOO : UNSIGNED (8),     ! Matrix or array too large
;  207          0376  1     BAS$K_VIRARROPE : UNSIGNED (8),
;  208          0377  1     BAS$K_VIRARRDIS : UNSIGNED (8),
;  209          0378  1     BAS$K_SUBOUTRAN : UNSIGNED (8),
;  210          0379  1     BAS$K_FATSYSIO : UNSIGNED (8),
;  211          0380  1     BAS$K_DATTYPERR : UNSIGNED (8),
;  212          0381  1     BAS$K_TOOFEWARG : UNSIGNED (8),
;  213          0382  1     BAS$K_TOOMANARG : UNSIGNED (8),
;  214          0383  1     BAS$K_ARGDONMAT : UNSIGNED (8),
;  215          0384  1     BAS$K_FLOPOIERR : UNSIGNED (8),
;  216          0385  1     BAS$K_RECBUCLOC : UNSIGNED (8),
;  217          0386  1     BAS$K_ONEOR_TWO : UNSIGNED (8),
;  218          0387  1     BAS$K_NOTIMP : UNSIGNED (8);
;  219          0388  1
```

```
;  221     0389  1 GLOBAL ROUTINE BAS$FETCH_BFA (                    ! Fetch a value by descriptor
;  222     0390  1         DESCRIP : REF BLOCK [8, BYTE],            ! The descriptor
;  223     0391  1         VALUE : REF BLOCK [8, BYTE],              ! Where to put it
;  224     0392  1         INDEX1                                    ! First index
;  225     0393  1     ) : NOVALUE =
;  226     0394  1
;  227     0395  1 !++
;  228     0396  1 ! FUNCTIONAL DESCRIPTION:
;  229     0397  1 !
;  230     0398  1 !       Fetch a value from an array or virtual array.  The location in
;  231     0399  1 !       which to place the value is passed by descriptor.
;  232     0400  1 !
;  233     0401  1 ! FORMAL PARAMETERS:
;  234     0402  1 !
;  235     0403  1 !       DESCRIP.rx.da   The descriptor of the array or virtual array
;  236     0404  1 !       VALUE.wx.dx     The place to put the value fetched
;  237     0405  1 !       INDEX1.rl.v     The first index into the array.  More indicies
;  238     0406  1 !                       may follow this one in the calling sequence.
;  239     0407  1 !
;  240     0408  1 ! IMPLICIT INPUTS:
;  241     0409  1 !
;  242     0410  1 !       NONE
;  243     0411  1 !
;  244     0412  1 ! IMPLICIT OUTPUTS:
;  245     0413  1 !
;  246     0414  1 !       NONE
;  247     0415  1 !
;  248     0416  1 ! ROUTINE VALUE:
;  249     0417  1 ! COMPLETION CODES:
;  250     0418  1 !
;  251     0419  1 !       NONE
;  252     0420  1 !
;  253     0421  1 ! SIDE EFFECTS:
;  254     0422  1 !
;  255     0423  1 !       Signals if an error is encountered.
;  256     0424  1 !
;  257     0425  1 !--
;  258     0426  1
;  259     0427  2     BEGIN
;  260     0428  2
;  261     0429  2     BUILTIN
;  262     0430  2         ACTUALCOUNT,
;  263     0431  2         ACTUALPARAMETER,
;  264     0432  2         ASHP;
;  265     0433  2
;  266     0434  2     LOCAL
;  267     0435  2         INDEX_VALUE,
;  268     0436  2         VALUE_LOCATION,
;  269     0437  2         MULTIPLIERS : REF VECTOR,
;  270     0438  2         BOUNDS : REF VECTOR,
;  271     0439  2         LOW_INDEX,
;  272     0440  2         HIGH_INDEX,
;  273     0441  2         INDEX_INCR,
;  274     0442  2         INDEX_NUMBER,
;  275     0443  2         VALUE_DESCR : BLOCK [12, BYTE],
;  276     0444  2         VALUE_DESC_ADDR,
;  277     0445  2         LENGTH;
```

```
;  278          0446  2
;  279          0447  2 !+
;  280          0448  2 ! Be sure the number of array subscripts matches the number of
;  281          0449  2 ! indicies given to us.
;  282          0450  2 !-
;  283          0451  2
;  284          0452  3     IF ((ACTUALCOUNT () - 2) NEQU .DESCRIP [DSC$B_DIMCT])
;  285          0453  3     THEN
;  286          0454  3         BEGIN
;  287          0455  3
;  288          0456  4         IF ((ACTUALCOUNT () - 2) LSSU .DESCRIP [DSC$B_DIMCT])
;  289          0457  4         THEN
;  290          0458  4             BAS$$STOP (BAS$K_TOOFEWARG)
;  291          0459  4         ELSE
;  292          0460  4             BAS$$STOP (BAS$K_TOOMANARG);
;  293          0461  4
;  294          0462  4         END;
;  295          0463  3
;  296          0464  2 !+
;  297          0465  2 ! The coefficients and bounds must be present.
;  298          0466  2 !-
;  299          0467  2
;  300          0468  2     IF ( NOT (.DESCRIP [DSC$V_FL_COEFF] AND .DESCRIP [DSC$V_FL_BOUNDS])) THEN BAS$$STOP (BAS$K_ARGDONMAT);
;  301          0469  2
;  302          0470  2     MULTIPLIERS = DESCRIP [DSC$L_M1];
;  303          0471  2     BOUNDS = DESCRIP [DSC$L_M1] + (%UPVAL*.DESCRIP [DSC$B_DIMCT]);
;  304          0472  2 !+
;  305          0473  2 ! Compute the lower and upper index numbers based on how the array
;  306          0474  2 ! is stored.
;  307          0475  2 !-
;  308          0476  2
;  309          0477  3     IF (.DESCRIP [DSC$V_FL_COLUMN])
;  310          0478  2     THEN
;  311          0479  3         BEGIN
;  312          0480  3         LOW_INDEX = .DESCRIP [DSC$B_DIMCT];
;  313          0481  3         HIGH_INDEX = 1;
;  314          0482  3         INDEX_INCR = -1;
;  315          0483  3         END
;  316          0484  2     ELSE
;  317          0485  3         BEGIN
;  318          0486  3         LOW_INDEX = 1;
;  319          0487  3         HIGH_INDEX = .DESCRIP [DSC$B_DIMCT];
;  320          0488  3         INDEX_INCR = 1;
;  321          0489  2         END;
;  322          0490  2
;  323          0491  2     INDEX_NUMBER = .LOW_INDEX - .INDEX_INCR;
;  324          0492  2 !+
;  325          0493  2 ! If this is a decimal array, the length is the number of 4 bit digits
;  326          0494  2 ! (not including the sign).  Convert this to the number of bytes.
;  327          0495  2 ! Decimal virtual arrays and record virtual arrays are stored with
;  328          0496  2 ! a length that is a multiple of 2 - check for that here also.
;  329          0497  2 !-
;  330          0498  2     CASE .DESCRIP [DSC$B_DTYPE] FROM DSC$K_DTYPE_Z TO DSC$K_DTYPE_H OF
;  331          0499  2         SET
;  332          0500  2
;  333          0501  2         [DSC$K_DTYPE_P]:                ! decimal
;  334          0502  3             BEGIN
```

```
;  335          0503  3              LENGTH = (.DESCRIP [DSC$W_LENGTH]/2) + 1;
;  336          0504  3              IF .DESCRIP [DSC$B_CLASS] EQL DSC$K_CLASS_BFA
;  337          0505  3              THEN
;  338          0506  4                  BEGIN
;  339          0507  4
;  340          0508  5                  LENGTH = ( INCR I FROM 1 TO 9 BY 1 DO
;  341          0509  6                                  IF .LENGTH LSS (1 ^ .I)
;  342          0510  4                                  THEN EXITLOOP (1 ^ .I) );
;  343          0511  4                  END;
;  344          0512  3              END;
;  345          0513  2
;  346          0514  2          [DSC$K_DTYPE_Z]:                ! record
;  347          0515  2              BEGIN
;  348          0516  2              LENGTH = .DESCRIP [DSC$W_LENGTH];
;  349          0517  2              IF .DESCRIP [DSC$B_CLASS] EQL DSC$K_CLASS_BFA
;  350          0518  2              THEN
;  351          0519  3                  BEGIN
;  352          0520  3
;  353          0521  4                  LENGTH = ( INCR I FROM 1 TO 9 BY 1 DO
;  354          0522  5                                  IF .LENGTH LSS (1 ^ .I)
;  355          0523  5                                  THEN EXITLOOP (1 ^ .I) );
;  356          0524  4                  END;
;  357          0525  3              END;
;  358          0526  2
;  359          0527  2          [INRANGE,OUTRANGE]:
;  360          0528  2              LENGTH = .DESCRIP [DSC$W_LENGTH];
;  361          0529  2          TES;
;  362          0530  2
;  363          0531  2      !+
;  364          0532  2      ! Compute the linear index from the indices provided.
;  365          0533  2      !-
;  366          0534  2          VALUE_LOCATION = 0;
;  367          0535  2
;  368          0536  2          WHILE ((INDEX_NUMBER = .INDEX_NUMBER + .INDEX_INCR) NEQ (.HIGH_INDEX + .INDEX_INCR)) DO
;  369          0537  3              BEGIN
;  370          0538  3              INDEX_VALUE = ACTUALPARAMETER (.INDEX_NUMBER + 2);
;  371          0539  3
;  372          0540  5              IF ((.INDEX_VALUE LSS .BOUNDS [(.INDEX_NUMBER - 1)*2])  !
;  373          0541  4                  OR (.INDEX_VALUE GTR .BOUNDS [((.INDEX_NUMBER - 1)*2) + 1]))
;  374          0542  3              THEN
;  375          0543  3                  BAS$$STOP (BAS$K_SUBOUTRAN);
;  376          0544  3
;  377          0545  3              VALUE_LOCATION = (.VALUE_LOCATION*.MULTIPLIERS [.INDEX_NUMBER - 1]) + .INDEX_VALUE;
;  378          0546  2              END;
;  379          0547  2
;  380          0548  2          VALUE_LOCATION = (.VALUE_LOCATION*.LENGTH) + .DESCRIP [DSC$A_A0];
;  381          0549  2      !+
;  382          0550  2      ! Build a descriptor pointing to the value cell in the array.  If this
;  383          0551  2      ! is an array of descriptors, the descriptor is copied, otherwise it
;  384          0552  2      ! is constructed.
;  385          0553  2      !-
;  386          0554  2
;  387          0555  3          IF (.DESCRIP [DSC$B_DTYPE] EQLU DSC$K_DTYPE_DSC)
;  388          0556  2          THEN
;  389          0557  3              BEGIN
;  390          0558  3
;  391          0559  3              MAP
```

```
;  392          0560  3            VALUE_LOCATION : REF BLOCK [8, BYTE];
;  393          0561  3
;  394          0562  3        VALUE_DESCR [DSC$W_LENGTH] = .VALUE_LOCATION [DSC$W_LENGTH];
;  395          0563  3        VALUE_DESCR [DSC$B_DTYPE] = .VALUE_LOCATION [DSC$B_DTYPE];
;  396          0564  3        VALUE_DESCR [DSC$B_CLASS] = (IF (.VALUE_LOCATION [DSC$B_CLASS] EQLU DSC$K_CLASS_D) THEN DSC$K_CLASS_
;  397          0565  3            ELSE .VALUE_LOCATION [DSC$B_CLASS]);
;  398          0566  3        VALUE_DESCR [DSC$A_POINTER] = .VALUE_LOCATION [DSC$A_POINTER];
;  399          0567  3        VALUE_DESC_ADDR = .VALUE_LOCATION;
;  400          0568  3        IF .VALUE_DESCR [DSC$B_DTYPE] EQL DSC$K_DTYPE_P
;  401          0569  3        THEN
;  402          0570  3            BEGIN
;  403          0571  4            MAP
;  404          0572  4                VALUE_LOCATION : REF BLOCK [12,BYTE];
;  405          0573  4            VALUE_DESCR [DSC$B_SCALE] = .VALUE_LOCATION [DSC$B_SCALE];
;  406          0574  4            END;
;  407          0575  3        END
;  408          0576  2    ELSE
;  409          0577  2        BEGIN
;  410          0578  3        VALUE_DESCR [DSC$W_LENGTH] = .DESCRIP [DSC$W_LENGTH];
;  411          0579  3        VALUE_DESCR [DSC$B_DTYPE] = .DESCRIP [DSC$B_DTYPE];
;  412          0580  3        VALUE_DESCR [DSC$B_CLASS] = DSC$K_CLASS_S;
;  413          0581  3        VALUE_DESCR [DSC$A_POINTER] = .VALUE_LOCATION;
;  414          0582  3        VALUE_DESC_ADDR = VALUE_DESCR;
;  415          0583  3        IF .VALUE_DESCR [DSC$B_DTYPE] EQL DSC$K_DTYPE_P
;  416          0584  3        THEN
;  417          0585  3            BEGIN
;  418          0586  4            MAP
;  419          0587  4                DESCRIP : REF BLOCK [12,BYTE];
;  420          0588  4            VALUE_DESCR [DSC$B_SCALE] = .DESCRIP [DSC$B_SCALE];
;  421          0589  4            END;
;  422          0590  3        END;
;  423          0591  2
;  424          0592  2    !+
;  425          0593  2    ! Make sure that the data type of the array element agrees with
;  426          0594  2    ! that of the result descriptor.
;  427          0595  2    !-
;  428          0596  2
;  429          0597  2        IF (.VALUE_DESCR [DSC$B_DTYPE] NEQU .VALUE [DSC$B_DTYPE]) THEN BAS$$STOP (BAS$K_DATTYPERR);
;  430          0598  2
;  431          0599  2    !+
;  432          0600  2    ! Special handling if this is a virtual array.
;  433          0601  2    !-
;  434          0602  2
;  435          0603  2        IF (.DESCRIP [DSC$B_CLASS] EQLU DSC$K_CLASS_BFA)
;  436          0604  2        THEN
;  437          0605  3            BEGIN
;  438          0606  3
;  439          0607  3            IF (.DESCRIP [DSC$B_DTYPE] EQLU DSC$K_DTYPE_DSC) THEN BAS$$STOP (BAS$K_NOTIMP);
;  440          0608  3
;  441          0609  3    !+
;  442          0610  3    ! If this is a string, remove trailing NULs when fetching it.
;  443          0611  3    !-
;  444          0612  3
;  445          0613  3            IF (.DESCRIP [DSC$B_DTYPE] EQL DSC$K_DTYPE_T)
;  446          0614  3            THEN
;  447          0615  4                BEGIN
;  448          0616  4
```

```
;  449          0617  4              LOCAL
;  450          0618  4                  TEMP_DESC : BLOCK [8, BYTE],
;  451          0619  4                  DATA_BUF : REF VECTOR [65535, BYTE],
;  452          0620  4                  LEN;
;  453          0621  4
;  454          0622  4              TEMP_DESC [DSC$W_LENGTH] = 0;
;  455          0623  4              TEMP_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  456          0624  4              TEMP_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  457          0625  4              TEMP_DESC [DSC$A_POINTER] = 0;
;  458          0626  4              STR$GET1_DX (DESCRIP [DSC$W_LENGTH], TEMP_DESC);
;  459          0627  4              BAS$$VA_FETCH (.DESCRIP, .VALUE_LOCATION, .TEMP_DESC [DSC$A_POINTER]);
;  460          0628  4      !+
;  461          0629  4      ! Now that we have it in temporary storage, remove its trailing NULs
;  462          0630  4      !-
;  463          0631  4              LEN = .TEMP_DESC [DSC$W_LENGTH];
;  464          0632  4              DATA_BUF = .TEMP_DESC [DSC$A_POINTER];
;  465          0633  4
;  466          0634  4              WHILE ((.LEN GTR 0) AND (.DATA_BUF [.LEN - 1] EQL 0)) DO
;  467          0635  4                  LEN = .LEN - 1;
;  468          0636  4
;  469          0637  4      !+
;  470          0638  4      ! Send the shortened string to the user.
;  471          0639  4      !-
;  472          0640  4              STR$COPY_R (.VALUE, LEN, .DATA_BUF);
;  473          0641  4      !+
;  474          0642  4      ! Deallocate our temporary string.
;  475          0643  4      !-
;  476          0644  4              STR$FREE1_DX (TEMP_DESC);
;  477          0645  4              END
;  478          0646  3          ELSE
;  479          0647  4              BEGIN
;  480          0648  4              IF .DESCRIP [DSC$B_DTYPE] NEQ DSC$K_DTYPE_P
;  481          0649  4              THEN
;  482          0650  4                  BAS$$VA_FETCH (.DESCRIP, .VALUE_LOCATION, .VALUE [DSC$A_POINTER])
;  483          0651  4              ELSE
;  484          0652  4                  BAS$$VA_FETCH (.DESCRIP, .VALUE_LOCATION, .VALUE);
;  485          0653  4              END
;  486          0654  3          END
;  487          0655  2      ELSE
;  488          0656  3          BEGIN
;  489          0657  3
;  490          0658  3          IF (.DESCRIP [DSC$B_CLASS] NEQU DSC$K_CLASS_A) THEN BAS$$STOP (BAS$K_NOTIMP);
;  491          0659  3
;  492          0660  3      !+
;  493          0661  3      ! Copy the array element to the value cell.  The form of the copy is
;  494          0662  3      ! based on the type of data.
;  495          0663  3      !-
;  496          0664  3
;  497          0665  3          CASE .VALUE [DSC$B_DTYPE] FROM DSC$K_DTYPE_Z TO DSC$K_DTYPE_H OF
;  498          0666  3              SET
;  499          0667  3
;  500          0668  3              [DSC$K_DTYPE_B] :                       ! byte
;  501          0669  3                  BLOCK [.VALUE [DSC$A_POINTER], 0, 0, %BPUNIT, 1]
;  502          0670  3                  = .BLOCK [.VALUE_DESCR [DSC$A_POINTER], 0, 0, %BPUNIT, 1];
;  503          0671  3
;  504          0672  3              [DSC$K_DTYPE_W] :                       ! 16-bit word
;  505          0673  3                  BLOCK [.VALUE [DSC$A_POINTER], 0, 0, %BPVAL/2, 1]       !
```

```
;  506     0674  3                = .BLOCK [.VALUE_DESCR [DSC$A_POINTER], 0, 0, %BPVAL/2, 1];
;  507     0675
;  508     0676                  [DSC$K_DTYPE_L] :                   ! 32-bit longword
;  509     0677                      BLOCK [.VALUE [DSC$A_POINTER], 0, 0, %BPVAL, 1] !
;  510     0678                      = .BLOCK [.VALUE_DESCR [DSC$A_POINTER], 0, 0, %BPVAL, 1];
;  511     0679
;  512     0680                  [DSC$K_DTYPE_F] :                   ! 32-bit floating point
;  513     0681                      BAS$$COPY_F_R1 (.VALUE_DESCR [DSC$A_POINTER], .VALUE [DSC$A_POINTER]);
;  514     0682
;  515     0683                  [DSC$K_DTYPE_D] :                   ! 64-bit double floating
;  516     0684                      BAS$$COPY_D_R1 (.VALUE_DESCR [DSC$A_POINTER], .VALUE [DSC$A_POINTER]);
;  517     0685
;  518     0686                  [DSC$K_DTYPE_G] :                   ! G floating
;  519     0687                      BAS$$COPY_G_R1 (.VALUE_DESCR [DSC$A_POINTER], .VALUE [DSC$A_POINTER]);
;  520     0688
;  521     0689                  [DSC$K_DTYPE_H] :                   ! H floating
;  522     0690                      BAS$$COPY_H_R3 (.VALUE_DESCR [DSC$A_POINTER], .VALUE [DSC$A_POINTER]);
;  523     0691
;  524     0692                  [DSC$K_DTYPE_T, DSC$K_DTYPE_Z] :    ! Text string or record
;  525     0693                      STR$COPY_DX (.VALUE, .VALUE_DESC_ADDR);
;  526     0694
;  527     0695                  [DSC$K_DTYPE_P] :                   ! decimal
;  528     0696                      BEGIN
;  529     0697  4                   LOCAL
;  530     0698  4                       COUNT;
;  531     0699  4                   COUNT = .VALUE_DESCR [DSC$B_SCALE] - .VALUE [DSC$B_SCALE];
;  532     0700  4                   ASHP (COUNT, VALUE_DESCR [DSC$W_LENGTH],
;  533     0701  4                       .VALUE_DESCR [DSC$A_POINTER], %REF(0), VALUE [DSC$W_LENGTH],
;  534     0702  4                       .VALUE [DSC$A_POINTER]);
;  535     0703                      END;
;  536     0704
;  537     0705                  [INRANGE, OUTRANGE] :
;  538     0706                      BAS$$STOP (BAS$K_DATTYPERR);
;  539     0707
;  540     0708                  TES;
;  541     0709
;  542     0710              END;
;  543     0711
;  544     0712  1       END;                                          ! end of BAS$FETCH_BFA


                                .TITLE  BAS$VIRTUAL_ARR
                                .IDENT  \1-033\

                                .EXTRN  BAS$$STOP, BAS$$COPY_F_R1
                                .EXTRN  BAS$$COPY_D_R1, BAS$$COPY_G_R1
                                .EXTRN  BAS$$COPY_H_R3, BAS$$VA_FETCH
                                .EXTRN  BAS$$VA_STORE, STR$GET1_DX
                                .EXTRN  STR$FREE1_DX, STR$COPY_DX
                                .EXTRN  STR$COPY_R, STR$CONCAT
                                .EXTRN  STR$DUPL_CHAR, BAS$K_MATARRTOO
                                .EXTRN  BAS$K_VIRARROPE
                                .EXTRN  BAS$K_VIRARRDIS
                                .EXTRN  BAS$K_SUBOUTRAN
                                .EXTRN  BAS$K_FATSYSIO
                                .EXTRN  BAS$K_DATTYPERR
                                .EXTRN  BAS$K_TOOFEWARG
```

```
                                                        .EXTRN  BAS$K_TOOMANARG
                                                        .EXTRN  BAS$K_ARGDONMAT
                                                        .EXTRN  BAS$K_FLOPOIERR
                                                        .EXTRN  BAS$K_RECBUCLOC
                                                        .EXTRN  BAS$K_ONEOR_TWO
                                                        .EXTRN  BAS$K_NOTIMP

                                                        .PSECT  _BAS$CODE,NOWRT,  SHR,  PIC,2

                                    0FFC 00000          .ENTRY  BAS$FETCH_BFA, Save R2,R3,R4,R5,R6,R7,R8,-      ; 0389
                                                                R9,R10,R11
                                5E          1C  C2 00002        SUBL2   #28, SP
                                50          6C  9A 00005        MOVZBL  (AP), R0                                ; 0452
                                50          02  C2 00008        SUBL2   #2, R0
                                56      04  AC  D0 0000B        MOVL    DESCRIP, R6
                                52      0B  A6  9A 0000F        MOVZBL  11(R6), R2
                                52          50  D1 00013        CMPL    R0, R2
                                            1C  13 00016        BEQL    3$
                                50          6C  9A 00018        MOVZBL  (AP), R0                                ; 0456
                                50          02  C2 0001B        SUBL2   #2, R0
                                52          50  D1 0001E        CMPL    R0, R2
                                            06  1E 00021        BGEQU   1$
                                7E      00G 8F  9A 00023        MOVZBL  #BAS$K_TOOFEWARG, -(SP)                 ; 0458
                                            04  11 00027        BRB     2$
                                7E      00G 8F  9A 00029 1$:    MOVZBL  #BAS$K_TOOMANARG, -(SP)                 ; 0460
                    00000000G   00          01  FB 0002D 2$:    CALLS   #1, BAS$$STOP
                05      0A  A6              06  E1 00034 3$:    BBC     #6, 10(R6), 4$                          ; 0468
                                0A  A6      95  00039           TSTB    10(R6)
                                            0B  19 0003C        BLSS    5$
                                7E      00G 8F  9A 0003E 4$:    MOVZBL  #BAS$K_ARGDONMAT, -(SP)
                    00000000G   00          01  FB 00042        CALLS   #1, BAS$$STOP
                                55      14  A6  9E 00049 5$:    MOVAB   20(R6), MULTIPLIERS                     ; 0470
                                57      14 A642 DE 0004D        MOVAL   20(R6)[R2], BOUNDS                      ; 0471
                08      0A  A6              05  E1 00052        BBC     #5, 10(R6), 6$                          ; 0477
                                51              52  D0 00057    MOVL    R2, LOW_INDEX                           ; 0480
                                50              01  D0 0005A    MOVL    #1, HIGH_INDEX                          ; 0481
                                5A              01  CE 0005D    MNEGL   #1, INDEX_INCR                          ; 0482
                                                09  11 00060    BRB     7$                                      ; 0477
                                51              01  D0 00062 6$:    MOVL    #1, LOW_INDEX                       ; 0486
                                50              52  D0 00065    MOVL    R2, HIGH_INDEX                          ; 0487
                                5A              01  D0 00068    MOVL    #1, INDEX_INCR                          ; 0488
                53              51              5A  C3 0006B 7$:    SUBL3   INDEX_INCR, LOW_INDEX, INDEX_NUMBER ; 0491
                                6E          02  A6  9A 0006F    MOVZBL  2(R6), (SP)                             ; 0498
                            1C  00              6E  8F 00073    CASEB   (SP), #0, #28
        003A            003A            003A        0066        00077 8$:   .WORD   12$-8$,-
        003A            003A            003A        003A        0007F       9$-8$,-
        003A            003A            003A        003A        00087       9$-8$,-
        003A            003A            003A        003A        0008F       9$-8$,-
        003A            003A            003A        003A        00097       9$-8$,-
        003A            003A            0042        003A        0009F       9$-8$,-
        003A            003A            003A        003A        000A7       9$-8$,-
                                                    003A        000AF       9$-8$,-
                                                                            9$-8$,-
                                                                            9$-8$,-
                                                                            9$-8$,-
                                                                            9$-8$,-
```

```
                                                              9$-8$,-                               ;
                                                              9$-8$,-                               ;
                                                              9$-8$,-                               ;
                                                              9$-8$,-                               ;
                                                              9$-8$,-                               ;
                                                              9$-8$,-                               ;
                                                              9$-8$,-                               ;
                                                              9$-8$,-                               ;
                                                              10$-8$,-                              ;
                                                              9$-8$,-                               ;
                                                              9$-8$,-                               ;
                                                              9$-8$,-                               ;
                                                              9$-8$,-                               ;
                                                              9$-8$,-                               ;
                                                              9$-8$,-                               ;
                                                              9$-8$                                 ;
                58           66  3C 000B1 9$:     MOVZWL      (R6), R8                              ; 0528
                54           58  D0 000B4         MOVL        R8, LENGTH                            ;
                             49  11 000B7         BRB         17$                                   ;
                58           66  3C 000B9 10$:    MOVZWL      (R6), R8                              ; 0503
51              58           02  C7 000BC         DIVL3       #2, R8, R1                            ;
                54       01  A1  9E 000C0         MOVAB       1(R1), LENGTH                         ;
          BF    8F       03  A6  91 000C4         CMPB        3(R6), #191                           ; 0504
                             37  12 000C9         BNEQ        17$                                   ;
                52           01  D0 000CB         MOVL        #1, I                                 ; 0509
51              01           52  78 000CE 11$:    ASHL        I, #1, R1                             ;
                51           54  D1 000D2         CMPL        LENGTH, R1                            ;
                             1F  19 000D5         BLSS        14$                                   ;
F3              52           09  F3 000D7         AOBLEQ      #9, I, 11$                            ;
                             22  11 000DB         BRB         16$                                   ; 0508
                58           66  3C 000DD 12$:    MOVZWL      (R6), R8                              ; 0516
                54           58  D0 000E0         MOVL        R8, LENGTH                            ;
          BF    8F       03  A6  91 000E3         CMPB        3(R6), #191                           ; 0517
                             18  12 000E8         BNEQ        17$                                   ;
                52           01  D0 000EA         MOVL        #1, I                                 ; 0522
51              01           52  78 000ED 13$:    ASHL        I, #1, R1                             ;
                51           54  D1 000F1         CMPL        LENGTH, R1                            ;
                             05  18 000F4         BGEQ        15$                                   ;
                54           51  D0 000F6 14$:    MOVL        R1, LENGTH                            ; 0523
                             07  11 000F9         BRB         17$                                   ;
EE              52           09  F3 000FB 15$:    AOBLEQ      #9, I, 13$                            ; 0522
                54           01  CE 000FF 16$:    MNEGL       #1, LENGTH                            ; 0521
                             52  D4 00102 17$:    CLRL        VALUE_LOCATION                        ; 0534
5B              50           5A  C1 00104         ADDL3       INDEX_INCR, HIGH_INDEX, R11           ; 0536
                53           5A  C0 00108 18$:    ADDL2       INDEX_INCR, INDEX_NUMBER              ;
                5B           53  D1 0010B         CMPL        INDEX_NUMBER, R11                     ;
                             2E  13 0010E         BEQL        21$                                   ;
                59    08 AC43    D0 00110         MOVL        8(AP)[INDEX_NUMBER], INDEX_VALUE      ; 0538
50              53           01  78 00115         ASHL        #1, INDEX_NUMBER, R0                  ; 0540
          F8 A740            59  D1 00119         CMPL        INDEX_VALUE, -8(BOUNDS)[R0]           ;
                             07  19 0011E         BLSS        19$                                   ;
          FC A740            59  D1 00120         CMPL        INDEX_VALUE, -4(BOUNDS)[R0]           ; 0541
                             0B  15 00125         BLEQ        20$                                   ;
                7E      00G  8F  9A 00127 19$:    MOVZBL      #BAS$K_SUBOUTRAN, -(SP)               ; 0543
     00000000G  00           01  FB 0012B         CALLS       #1, BAS$$STOP                         ;
50              52   FC A543     C5 00132 20$:    MULL3       -4(MULTIPLIERS)[INDEX_NUMBER], -      ; 0545
                                                              VALUE_LOCATION, R0                    ;
```

```
                52              50              59  C1  00138          ADDL3   INDEX_VALUE, R0, VALUE_LOCATION                 ;
                                                CA  11  0013C          BRB     18$                                             ; 0536
                50              52              54  C5  0013E  21$:    MULL3   LENGTH, VALUE_LOCATION, R0                      ; 0548
                52              50          10  A6  C1  00142          ADDL3   16(R6), R0, VALUE_LOCATION                    ;
                                                57  D4  00147          CLRL    R7                                              ; 0555
                                18              6E  91  00149          CMPB    (SP), #24                                       ;
                                                33  12  0014C          BNEQ    24$                                             ;
                                                57  D6  0014E          INCL    R7                                              ;
                        10      AE              62  B0  00150          MOVW    (VALUE_LOCATION), VALUE_DESCR                   ; 0562
                        12      AE          02  A2  90  00154          MOVB    2(VALUE_LOCATION), VALUE_DESCR+2                ; 0563
                                02          03  A2  91  00159          CMPB    3(VALUE_LOCATION), #2                           ; 0564
                                                05  12  0015D          BNEQ    22$                                             ;
                                50              01  D0  0015F          MOVL    #1, R0                                          ;
                                                04  11  00162          BRB     23$                                             ;
                                50          03  A2  9A  00164  22$:    MOVZBL  3(VALUE_LOCATION), R0                           ; 0565
                        13      AE              50  90  00168  23$:    MOVB    R0, VALUE_DESCR+3                               ; 0564
                        14      AE          04  A2  D0  0016C          MOVL    4(VALUE_LOCATION), VALUE_DESCR+4                ; 0566
                                54              52  D0  00171          MOVL    VALUE_LOCATION, VALUE_DESC_ADDR                 ; 0567
                                15          12  AE  91  00174          CMPB    VALUE_DESCR+2, #21                              ; 0568
                                                26  12  00178          BNEQ    25$                                             ;
                        18      AE          08  A2  90  0017A          MOVB    8(VALUE_LOCATION), VALUE_DESCR+8                ; 0573
                                                1F  11  0017F          BRB     25$                                             ; 0555
                        10      AE              58  B0  00181  24$:    MOVW    R8, VALUE_DESCR                                 ; 0578
                        12      AE              6E  90  00185          MOVB    (SP), VALUE_DESCR+2                             ; 0579
                        13      AE              01  90  00189          MOVB    #1, VALUE_DESCR+3                               ; 0580
                        14      AE              52  D0  0018D          MOVL    VALUE_LOCATION, VALUE_DESCR+4                   ; 0581
                                54          10  AE  9E  00191          MOVAB   VALUE_DESCR, VALUE_DESC_ADDR                    ; 0582
                                15          12  AE  91  00195          CMPB    VALUE_DESCR+2, #21                              ; 0583
                                                05  12  00199          BNEQ    25$                                             ;
                        18      AE          08  A6  90  0019B          MOVB    8(R6), VALUE_DESCR+8                            ; 0588
                                55          08  AC  D0  001A0  25$:    MOVL    VALUE, R5                                       ; 0597
                        02      A5          12  AE  91  001A4          CMPB    VALUE_DESCR+2, 2(R5)                            ;
                                                0B  13  001A9          BEQL    26$                                             ;
                                7E      00G     8F  9A  001AB          MOVZBL  #BAS$K_DATTYPERR, -(SP)                         ;
                00000000G       00              01  FB  001AF          CALLS   #1, BAS$$STOP                                   ;
                        BF      8F          03  A6  91  001B6  26$:    CMPB    3(R6), #191                                     ; 0603
                                                03  13  001BB          BEQL    27$                                             ;
                                                    0086  31  001BD    BRW     34$                                             ;
                                0B                  57  E9  001C0  27$:    BLBC    R7, 28$                                 ; 0607
                                7E      00G         8F  9A  001C3          MOVZBL  #BAS$K_NOTIMP, -(SP)                    ;
                00000000G       00                  01  FB  001C7          CALLS   #1, BAS$$STOP                           ;
                                0E                  6E  91  001CE  28$:    CMPB    (SP), #14                               ; 0613
                                                    5B  12  001D1          BNEQ    31$                                     ;
                        0B      AE  020E0000        8F  D0  001D3          MOVL    #34471936, TEMP_DESC                    ; 0622
                                                0C  AE  D4  001DB          CLRL    TEMP_DESC+4                             ; 0625
                                                08  AE  9F  001DE          PUSHAB  TEMP_DESC                               ; 0626
                                                    56  DD  001E1          PUSHL   R6                                      ;
                00000000G       00                  02  FB  001E3          CALLS   #2, STR$GET1_DX                         ;
                                                0C  AE  DD  001EA          PUSHL   TEMP_DESC+4                             ; 0627
                                                    52  DD  001ED          PUSHL   VALUE_LOCATION                          ;
                                                    56  DD  001EF          PUSHL   R6                                      ;
                00000000G       00                  03  FB  001F1          CALLS   #3, BAS$$VA_FETCH                       ;
                        04      AE          08  AE  3C  001F8          MOVZWL  TEMP_DESC, LEN                                  ; 0631
                                50          0C  AE  D0  001FD          MOVL    TEMP_DESC+4, DATA_BUF                           ; 0632
                                            04  AE  D5  00201  29$:    TSTL    LEN                                             ; 0634
                                                0F  15  00204          BLEQ    30$                                             ;
        51                      50          04  AE  C1  00206          ADDL3   LEN, DATA_BUF, R1                               ;
```

```
                                FF    A1  95 0020B          TSTB    -1(R1)                                                          ;
                                      05  12 0020E          BNEQ    30$                                                             ;
                                04    AE  D7 00210          DECL    LEN                                                             ; 0635
                                      EC  11 00213          BRB     29$                                                             ;
                                      50  DD 00215 30$:     PUSHL   DATA_BUF                                                        ; 0640
                                08    AE  9F 00217          PUSHAB  LEN                                                             ;
                                      55  DD 0021A          PUSHL   R5                                                              ;
                    00000000G   00        03  FB 0021C     CALLS   #3, STR$COPY_R                                                  ;
                                08    AE  9F 00223          PUSHAB  TEMP_DESC                                                       ; 0644
                    00000000G   00        01  FB 00226     CALLS   #1, STR$FREE1_DX                                                ;
                                          04  0022D         RET                                                                     ; 0613
                                15        6E  91 0022E 31$:  CMPB    (SP), #21                                                    ; 0648
                                          05  13 00231      BEQL    32$                                                             ;
                                04        A5  DD 00233      PUSHL   4(R5)                                                           ; 0650
                                          02  11 00236      BRB     33$                                                             ;
                                          55  DD 00238 32$:  PUSHL   R5                                                            ; 0652
                                          52  DD 0023A 33$:  PUSHL   VALUE_LOCATION                                                 ;
                                          56  DD 0023C      PUSHL   R6                                                              ;
                    00000000G   00        03  FB 0023E     CALLS   #3, BAS$$VA_FETCH                                               ;
                                          04  00245         RET                                                                     ; 0613
                                04        03  A6  91 00246 34$:  CMPB    3(R6), #4                                                ; 0658
                                          0B  13 0024A      BEQL    35$                                                             ;
                                      7E  00G 8F  9A 0024C      MOVZBL  #BAS$K_NOTIMP, -(SP)                                        ;
                    00000000G   00        01  FB 00250     CALLS   #1, BAS$$STOP                                                   ;
                          1C            02  A5  8F 00257 35$:  CASEB   2(R5), #0, #28                                            ; 0665
          003A        003A        003A        0094    0025C 36$:  .WORD   45$-36$,-                                                 ;
          004C        0046        003A        003A    00264       37$-36$,-                                                         ;
          0067        0058        003A        0052    0026C       37$-36$,-                                                         ;
          003A        0094        003A        003A    00274       37$-36$,-                                                         ;
          003A        003A        003A        003A    0027C       37$-36$,-                                                         ;
          003A        003A        00A0        003A    00284       37$-36$,-                                                         ;
          0076        003A        003A        003A    0028C       38$-36$,-                                                         ;
                                              0085    00294       39$-36$,-                                                         ;
                                                                  40$-36$,-                                                         ;
                                                                  37$-36$,-                                                         ;
                                                                  41$-36$,-                                                         ;
                                                                  42$-36$,-                                                         ;
                                                                  37$-36$,-                                                         ;
                                                                  37$-36$,-                                                         ;
                                                                  45$-36$,-                                                         ;
                                                                  37$-36$,-                                                         ;
                                                                  37$-36$,-                                                         ;
                                                                  37$-36$,-                                                         ;
                                                                  37$-36$,-                                                         ;
                                                                  37$-36$,-                                                         ;
                                                                  37$-36$,-                                                         ;
                                                                  46$-36$,-                                                         ;
                                                                  37$-36$,-                                                         ;
                                                                  37$-36$,-                                                         ;
                                                                  37$-36$,-                                                         ;
                                                                  37$-36$,-                                                         ;
                                                                  37$-36$,-                                                         ;
                                                                  43$-36$,-                                                         ;
                                                                  44$-36$                                                           ;
                                      7E  00G 8F  9A 00296 37$:  MOVZBL  #BAS$K_DATTYPERR, -(SP)                                    ; 0706
                    00000000G   00        01  FB 0029A     CALLS   #1, BAS$$STOP                                                   ;
                                          04  002A1         RET                                                                     ;
```

```
                          04   B5        14   BE  90 002A2 38$:    MOVB    @VALUE_DESCR+4, @4(R5)                          ; 0670
                                                 04 002A7         RET
                          04   B5        14   BE  B0 002A8 39$:    MOVW    @VALUE_DESCR+4, @4(R5)                          ; 0674
                                                 04 002AD         RET
                          04   B5        14   BE  D0 002AE 40$:    MOVL    @VALUE_DESCR+4, @4(R5)                          ; 0678
                                                 04 002B3         RET
                               51        04   A5  D0 002B4 41$:    MOVL    4(R5), R1                                       ; 0681
                               50        14   AE  D0 002B8         MOVL    VALUE_DESCR+4, R0
                                    00000000G   00  16 002BC         JSB     BAS$$COPY_F_R1
                                                 04 002C2         RET
                               51        04   A5  D0 002C3 42$:    MOVL    4(R5), R1                                       ; 0684
                               50        14   AE  D0 002C7         MOVL    VALUE_DESCR+4, R0
                                    00000000G   00  16 002CB         JSB     BAS$$COPY_D_R1
                                                 04 002D1         RET
                               51        04   A5  D0 002D2 43$:    MOVL    4(R5), R1                                       ; 0687
                               50        14   AE  D0 002D6         MOVL    VALUE_DESCR+4, R0
                                    00000000G   00  16 002DA         JSB     BAS$$COPY_G_R1
                                                 04 002E0         RET
                               51        04   A5  D0 002E1 44$:    MOVL    4(R5), R1                                       ; 0690
                               50        14   AE  D0 002E5         MOVL    VALUE_DESCR+4, R0
                                    00000000G   00  16 002E9         JSB     BAS$$COPY_H_R3
                                                 04 002EF         RET
                                              54  DD 002F0 45$:    PUSHL   VALUE_DESC_ADDR                                 ; 0693
                                              55  DD 002F2         PUSHL   R5
                          00000000G   00          02  FB 002F4         CALLS   #2, STR$COPY_DX
                                                 04 002FB         RET
                               50        18   AE  98 002FC 46$:    CVTBL   VALUE_DESCR+8, COUNT                            ; 0699
                               51        08   A5  98 00300         CVTBL   8(R5), R1
                               50              51  C2 00304         SUBL2   R1, COUNT
          00        14   BE        10   AE        50  F8 00307         ASHP    COUNT, VALUE_DESCR, @VALUE_DESCR+4, #0, -       ; 0702
                               04   B5              65      0030E                 (R5), @4(R5)
                                                 04 00311         RET                                                     ; 0712

; Routine Size:  786 bytes,    Routine Base:  _BAS$CODE + 0000

;  545          0713  1
```

```
;  547          0714  1 GLOBAL ROUTINE BAS$STORE_BFA (                        ! Store a value by descriptor
;  548          0715  1         VALUE : REF BLOCK [8, BYTE],                 ! Where to find the value
;  549          0716  1         DESCRIP : REF BLOCK [8, BYTE],               ! The descriptor to store it
;  550          0717  1         INDEX1                                       ! First index
;  551          0718  1     ) : NOVALUE =
;  552          0719  1
;  553          0720  1 !++
;  554          0721  1 ! FUNCTIONAL DESCRIPTION:
;  555          0722  1 !
;  556          0723  1 !       Store a value in an array or virtual array.  The location from
;  557          0724  1 !       which to fetch the value is passed by descriptor.
;  558          0725  1 !
;  559          0726  1 ! FORMAL PARAMETERS:
;  560          0727  1 !
;  561          0728  1 !       VALUE.rx.dx     The place from which to get the value stored
;  562          0729  1 !       DESCRIP.rx.da   The descriptor of the array or virtual array
;  563          0730  1 !       INDEX1.rl.v     The first index into the array.  More indicies
;  564          0731  1 !                       may follow this one in the calling sequence.
;  565          0732  1 !
;  566          0733  1 ! IMPLICIT INPUTS:
;  567          0734  1 !
;  568          0735  1 !       NONE
;  569          0736  1 !
;  570          0737  1 ! IMPLICIT OUTPUTS:
;  571          0738  1 !
;  572          0739  1 !       NONE
;  573          0740  1 !
;  574          0741  1 ! ROUTINE VALUE:
;  575          0742  1 ! COMPLETION CODES:
;  576          0743  1 !
;  577          0744  1 !       NONE
;  578          0745  1 !
;  579          0746  1 ! SIDE EFFECTS:
;               0747  1 !
;               0748  1 !       Signals if an error is encountered.
;               0749  1 !
;  583          0750  1 !--
;  584          0751  1
;  585          0752  2     BEGIN
;  586          0753  2
;  587          0754  2     BUILTIN
;  588          0755  2         ACTUALCOUNT,
;  589          0756  2         ACTUALPARAMETER,
;  590          0757  2         ASHP;
;  591          0758  2
;  592          0759  2     LOCAL
;  593          0760  2         INDEX_VALUE,
;  594          0761  2         VALUE_LOCATION,
;  595          0762  2         MULTIPLIERS : REF VECTOR,
;  596          0763  2         BOUNDS : REF VECTOR,
;  597          0764  2         LOW_INDEX,
;  598          0765  2         HIGH_INDEX,
;  599          0766  2         INDEX_INCR,
;  600          0767  2         INDEX_NUMBER,
;  601          0768  2         VALUE_DESCR : BLOCK [12, BYTE],
;  602          0769  2         VALUE_DESC_ADDR,
;  603          0770  2         LENGTH;
```

```
;  604          0771  2
;  605          0772  2        !+
;  606          0773  2        ! Be sure the number of array subscripts matches the number of
;  607          0774  2        ! indicies given to us.
;  608          0775  2        !-
;  609          0776  2
;  610          0777  2            IF ((ACTUALCOUNT () - 2) NEQU .DESCRIP [DSC$B_DIMCT])
;  611          0778  2            THEN
;  612          0779  2                BEGIN
;  613          0780  2
;  614          0781  2                IF ((ACTUALCOUNT () - 2) LSSU .DESCRIP [DSC$B_DIMCT])
;  615          0782  2                THEN
;  616          0783  2                    BAS$$STOP (BAS$K_TOOFEWARG)
;  617          0784  2                ELSE
;  618          0785  2                    BAS$$STOP (BAS$K_TOOMANARG);
;  619          0786  2
;  620          0787  2                END;
;  621          0788  2
;  622          0789  2        !+
;  623          0790  2        ! The coefficients and bounds must be present.
;  624          0791  2        !-
;  625          0792  2
;  626          0793  2            IF ( NOT (.DESCRIP [DSC$V_FL_COEFF] AND .DESCRIP [DSC$V_FL_BOUNDS])) THEN BAS$$STOP (BAS$K_ARGDONMAT);
;  627          0794  2
;  628          0795  2            MULTIPLIERS = DESCRIP [DSC$L_M1];
;  629          0796  2            BOUNDS = DESCRIP [DSC$L_M1] + (%UPVAL*.DESCRIP [DSC$B_DIMCT]);
;  630          0797  2        !+
;  631          0798  2        ! Compute the lower and upper index numbers based on how the array
;  632          0799  2        ! is stored.
;  633          0800  2        !-
;  634          0801  2
;  635          0802  2            IF (.DESCRIP [DSC$V_FL_COLUMN])
;  636          0803  2            THEN
;  637          0804  2                BEGIN
;  638          0805  2                LOW_INDEX = .DESCRIP [DSC$B_DIMCT];
;  639          0806  2                HIGH_INDEX = 1;
;  640          0807  2                INDEX_INCR = -1;
;  641          0808  2                END
;  642          0809  2            ELSE
;  643          0810  2                BEGIN
;  644          0811  2                LOW_INDEX = 1;
;  645          0812  2                HIGH_INDEX = .DESCRIP [DSC$B_DIMCT];
;  646          0813  2                INDEX_INCR = 1;
;  647          0814  2                END;
;  648          0815  2
;  649          0816  2            INDEX_NUMBER = .LOW_INDEX - .INDEX_INCR;
;  650          0817  2        !+
;  651          0818  2        ! If this is a decimal array, the length in the descriptor is the number of
;  652          0819  2        ! 4 bit digits (not including the sign).  Convert this length to the number
;  653          0820  2        ! of bytes.
;  654          0821  2        ! Also, if this is a virtual array, the size must be a multiple of 2.  This
;  655          0822  2        ! is true for arrays of records as well.
;  656          0823  2        !-
;  657          0824  2            CASE .DESCRIP [DSC$B_DTYPE] FROM DSC$K_DTYPE_Z TO DSC$K_DTYPE_H OF
;  658          0825  2                SET
;  659          0826  2
;  660          0827  2                [DSC$K_DTYPE_P] :               ! decimal
```

```
;  661          0828  3              BEGIN
;  662          0829  3              LENGTH = (.DESCRIP [DSC$W_LENGTH]/2) + 1;
;  663          0830  3              IF .DESCRIP [DSC$B_CLASS] EQL DSC$K_CLASS_BFA
;  664          0831  3              THEN
;  665          0832  4                  BEGIN
;  666          0833  4
;  667          0834  5                  LENGTH = ( INCR I FROM 1 TO 9 BY 1 DO
;  668          0835  6                                  IF .LENGTH LSS (1 ^ .I)
;  669          0836  4                                  THEN EXITLOOP (1 ^ .I) );
;  670          0837  3                  END;
;  671          0838  2              END;
;  672          0839  2
;  673          0840  2          [DSC$K_DTYPE_Z] :                  ! record
;  674          0841  3              BEGIN
;  675          0842  3              LENGTH = .DESCRIP [DSC$W_LENGTH];
;  676          0843  3              IF .DESCRIP [DSC$B_CLASS] EQL DSC$K_CLASS_BFA
;  677          0844  3              THEN
;  678          0845  4                  BEGIN
;  679          0846  4
;  680          0847  5                  LENGTH = ( INCR I FROM 1 TO 9 BY 1 DO
;  681          0848  6                                  IF .LENGTH LSS (1 ^ .I)
;  682          0849  4                                  THEN EXITLOOP (1 ^ .I) );
;  683          0850  3                  END;
;  684          0851  2              END;
;  685          0852  2
;  686          0853  2          [INRANGE, OUTRANGE] :
;  687          0854  2              LENGTH = .DESCRIP [DSC$W_LENGTH];
;  688          0855  2          TES;
;  689          0856  2
;  690          0857  2      !+
;  691          0858  2      ! Compute the linear index from the indices provided.
;  692          0859  2      !-
;  693          0860  2          VALUE_LOCATION = 0;
;  694          0861  2
;  695          0862  2          WHILE ((INDEX_NUMBER = .INDEX_NUMBER + .INDEX_INCR) NEQ (.HIGH_INDEX + .INDEX_INCR)) DO
;  696          0863  3              BEGIN
;  697          0864  3              INDEX_VALUE = ACTUALPARAMETER (.INDEX_NUMBER + 2);
;  698          0865  3
;  699          0866  3              IF ((.INDEX_VALUE LSS .BOUNDS [(.INDEX_NUMBER - 1)*2])  !
;  700          0867  4                  OR (.INDEX_VALUE GTR .BOUNDS [((.INDEX_NUMBER - 1)*2) + 1]))
;  701          0868  3              THEN
;  702          0869  4                  BAS$$STOP (BAS$K_SUBOUTRAN);
;  703          0870  3
;  704          0871  3              VALUE_LOCATION = (.VALUE_LOCATION*.MULTIPLIERS [.INDEX_NUMBER - 1]) + .INDEX_VALUE;
;  705          0872  3              END;
;  706          0873  2
;  707          0874  2          VALUE_LOCATION = (.VALUE_LOCATION*.LENGTH) + .DESCRIP [DSC$A_A0];
;  708          0875  2      !+
;  709          0876  2      ! Build a descriptor pointing to the value cell in the array.  If this
;  710          0877  2      ! is an array of descriptors, the descriptor is copied, otherwise it
;  711          0878  2      ! is constructed.
;  712          0879  2      !-
;  713          0880  2
;  714          0881  2          IF (.DESCRIP [DSC$B_DTYPE] EQLU DSC$K_DTYPE_DSC)
;  715          0882  2          THEN
;  716          0883  3              BEGIN
;  717          0884  3
```

```
;  718          0885  3          MAP
;  719          0886  3              VALUE_LOCATION : REF BLOCK [8, BYTE];
;  720          0887  3
;  721          0888  3          VALUE_DESCR [DSC$W_LENGTH] = .VALUE_LOCATION [DSC$W_LENGTH];
;  722          0889  3          VALUE_DESCR [DSC$B_DTYPE] = .VALUE_LOCATION [DSC$B_DTYPE];
;  723          0890  4          VALUE_DESCR [DSC$B_CLASS] = (IF (.VALUE_LOCATION [DSC$B_CLASS] EQLU DSC$K_CLASS_D) THEN DSC$K_CLASS_
;  724          0891  3              ELSE .VALUE_LOCATION [DSC$B_CLASS]);
;  725          0892  3          VALUE_DESCR [DSC$A_POINTER] = .VALUE_LOCATION [DSC$A_POINTER];
;  726          0893  3          VALUE_DESC_ADDR = .VALUE_LOCATION;
;  727          0894  3          IF .VALUE_DESCR [DSC$B_DTYPE] EQL DSC$K_DTYPE_P
;  728          0895  3          THEN
;  729          0896  4              BEGIN
;  730          0897  4              MAP
;  731          0898  4                  VALUE_LOCATION : REF BLOCK [12,BYTE];
;  732          0899  4              VALUE_DESCR [DSC$B_SCALE] = .VALUE_LOCATION [DSC$B_SCALE];
;  733          0900  3              END;
;  734          0901  3          END
;  735          0902  2      ELSE
;  736          0903  3          BEGIN
;  737          0904  3          VALUE_DESCR [DSC$W_LENGTH] = .DESCRIP [DSC$W_LENGTH];
;  738          0905  3          VALUE_DESCR [DSC$B_DTYPE] = .DESCRIP [DSC$B_DTYPE];
;  739          0906  3          VALUE_DESCR [DSC$B_CLASS] = DSC$K_CLASS_S;
;  740          0907  3          VALUE_DESCR [DSC$A_POINTER] = .VALUE_LOCATION;
;  741          0908  3          VALUE_DESC_ADDR = VALUE_DESCR;
;  742          0909  3          IF .VALUE_DESCR [DSC$B_DTYPE] EQL DSC$K_DTYPE_P
;  743          0910  3          THEN
;  744          0911  4              BEGIN
;  745          0912  4              MAP
;  746          0913  4                  DESCRIP : REF BLOCK [12,BYTE];
;  747          0914  4              VALUE_DESCR [DSC$B_SCALE] = .DESCRIP [DSC$B_SCALE];
;  748          0915  3              END;
;  749          0916  2          END;
;  750          0917  2
;  751          0918  2      !+
;  752          0919  2      ! Make sure that the data type of the array element agrees with
;  753          0920  2      ! that of the source descriptor.
;  754          0921  2      !-
;  755          0922  2
;  756          0923  2          IF (.VALUE_DESCR [DSC$B_DTYPE] NEQU .VALUE [DSC$B_DTYPE]) THEN BAS$$STOP (BAS$K_DATTYPERR);
;  757          0924  2
;  758          0925  2      !+
;  759          0926  2      ! Special handling if this is a virtual array.
;  760          0927  2      !-
;  761          0928  2
;  762          0929  3          IF (.DESCRIP [DSC$B_CLASS] EQLU DSC$K_CLASS_BFA)
;  763          0930  2          THEN
;  764          0931  3              BEGIN
;  765          0932  3
;  766          0933  3              IF (.DESCRIP [DSC$B_DTYPE] EQLU DSC$K_DTYPE_DSC) THEN BAS$$STOP (BAS$K_NOTIMP);
;  767          0934  3
;  768          0935  3      !+
;  769          0936  3      ! If this is a string, we must pad it with NULs.  To do this, we need
;  770          0937  3      ! a temporary string.
;  771          0938  3      !-
;  772          0939  3
;  773          0940  4              IF (.DESCRIP [DSC$B_DTYPE] EQL DSC$K_DTYPE_T)
;  774          0941  3              THEN
```

```
;  775          0942  4              BEGIN
;  776          0943  4
;  777          0944  4              LOCAL
;  778          0945  4                  NULLS_COUNT,
;  779          0946  4                  TEMP_DESC : BLOCK [8, BYTE];
;  780          0947  4
;  781          0948  4          !+
;  782          0949  4          ! Copy the caller's string to our temporary.
;  783          0950  4          !-
;  784          0951  4              TEMP_DESC [DSC$W_LENGTH] = 0;
;  785          0952  4              TEMP_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  786          0953  4              TEMP_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  787          0954  4              TEMP_DESC [DSC$A_POINTER] = 0;
;  788          0955  4              STR$COPY_DX (TEMP_DESC, .VALUE);
;  789          0956  4          !+
;  790          0957  4          ! Concatenate enough NULs onto the string to make it the right length.
;  791          0958  4          !-
;  792          0959  4              NULLS_COUNT = .DESCRIP [DSC$W_LENGTH] - .TEMP_DESC [DSC$W_LENGTH];
;  793          0960  4
;  794          0961  5              IF (.NULLS_COUNT GTR 0)
;  795          0962  4              THEN
;  796          0963  5                  BEGIN
;  797          0964  5
;  798          0965  5                  LOCAL
;  799          0966  5                      NULLS_DESC : BLOCK [8, BYTE];
;  800          0967  5
;  801          0968  5                  NULLS_DESC [DSC$W_LENGTH] = 1;
;  802          0969  5                  NULLS_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  803          0970  5                  NULLS_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  804          0971  5                  NULLS_DESC [DSC$A_POINTER] = 0;
;  805          0972  5                  STR$DUPL_CHAR (NULLS_DESC, %REF (.NULLS_COUNT), %REF (0));
;  806          0973  5                  STR$CONCAT (TEMP_DESC, TEMP_DESC, NULLS_DESC);
;  807          0974  5                  STR$FREE1_DX (NULLS_DESC);
;  808          0975  4                  END;
;  809          0976  4
;  810          0977  4          !+
;  811          0978  4          ! Now store the (possibly lengthened) string in the file.
;  812          0979  4          !-
;  813          0980  4              BAS$$VA_STORE (.DESCRIP, .VALUE_LOCATION, .TEMP_DESC [DSC$A_POINTER]);
;  814          0981  4          !+
;  815          0982  4          ! Free our temporary string.
;  816          0983  4          !-
;  817          0984  4              STR$FREE1_DX (TEMP_DESC);
;  818          0985  4              END
;  819          0986  3          ELSE
;  820          0987  4              BEGIN
;  821          0988  4              IF .DESCRIP [DSC$B_DTYPE] NEQ DSC$K_DTYPE_P
;  822          0989  4              THEN
;  823          0990  4                  BAS$$VA_STORE (.DESCRIP, .VALUE_LOCATION, .VALUE [DSC$A_POINTER])
;  824          0991  4              ELSE
;  825          0992  4                  BAS$$VA_STORE (.DESCRIP, .VALUE_LOCATION, .VALUE);
;  826          0993  4              END
;  827          0994  3          END
;  828          0995  2      ELSE
;  829          0996  3          BEGIN
;  830          0997  3
;  831          0998  3          IF (.DESCRIP [DSC$B_CLASS] NEQU DSC$K_CLASS_A) THEN BAS$$STOP (BAS$K_NOTIMP);
```

```
;  832          0999  3
;  833          1000  3     !+
;  834          1001  3     ! Copy the value cell to the array element.  The form of the copy is
;  835          1002  3     ! based on the type of data.
;  836          1003  3     !-
;  837          1004  3
;  838          1005  3             CASE .VALUE [DSC$B_DTYPE] FROM DSC$K_DTYPE_Z TO DSC$K_DTYPE_H OF
;  839          1006  3                 SET
;  840          1007  3
;  841          1008  3                 [DSC$K_DTYPE_B] :                   ! byte
;  842          1009  3                     BLOCK [.VALUE_DESCR [DSC$A_POINTER], 0, 0, %BPUNIT, 1]
;  843          1010  3                     = .BLOCK [.VALUE [DSC$A_POINTER], 0, 0, %BPUNIT, 1];
;  844          1011  3
;  845          1012  3                 [DSC$K_DTYPE_W] :                   ! 16-bit word
;  846          1013  3                     BLOCK [.VALUE_DESCR [DSC$A_POINTER], 0, 0, %BPVAL/2, 1] !
;  847          1014  3                     = .BLOCK [.VALUE [DSC$A_POINTER], 0, 0, %BPVAL/2, 1];
;  848          1015  3
;  849          1016  3                 [DSC$K_DTYPE_L] :                   ! 32-bit longword
;  850          1017  3                     BLOCK [.VALUE_DESCR [DSC$A_POINTER], 0, 0, %BPVAL, 1]   !
;  851          1018  3                     = .BLOCK [.VALUE [DSC$A_POINTER], 0, 0, %BPVAL, 1];
;  852          1019  3
;  853          1020  3                 [DSC$K_DTYPE_F] :                   ! 32-bit floating point
;  854          1021  3                     BAS$$COPY_F_R1 (.VALUE [DSC$A_POINTER], .VALUE_DESCR [DSC$A_POINTER]);
;  855          1022  3
;  856          1023  3                 [DSC$K_DTYPE_D] :                   ! 64-bit double floating
;  857          1024  3                     BAS$$COPY_D_R1 (.VALUE [DSC$A_POINTER], .VALUE_DESCR [DSC$A_POINTER]);
;  858          1025  3
;  859          1026  3                 [DSC$K_DTYPE_G] :                   ! G floating
;  860          1027  3                     BAS$$COPY_G_R1 (.VALUE [DSC$A_POINTER], .VALUE_DESCR [DSC$A_POINTER]);
;  861          1028  3
;  862          1029  3                 [DSC$K_DTYPE_H] :                   ! H floating
;  863          1030  3                     BAS$$COPY_H_R3 (.VALUE [DSC$A_POINTER], .VALUE_DESCR [DSC$A_POINTER]);
;  864          1031  3
;  865          1032  3                 [DSC$K_DTYPE_T, DSC$K_DTYPE_Z] :    ! Text string or record
;  866          1033  3                     STR$COPY_DX (.VALUE_DESC_ADDR, .VALUE);
;  867          1034  3
;  868          1035  3                 [DSC$K_DTYPE_P] :                   ! decimal
;  869          1036  4                     BEGIN
;  870          1037  4                     MAP
;  871          1038  4                         VALUE : REF BLOCK [12,BYTE];
;  872          1039  4                     LOCAL
;  873          1040  4                         COUNT;
;  874          1041  4                     COUNT = .VALUE [DSC$B_SCALE] - .VALUE_DESCR [DSC$B_SCALE];
;  875          1042  4                     ASHP (COUNT, VALUE [DSC$W_LENGTH], .VALUE [DSC$A_POINTER],
;  876          1043  4                         %REF(0), VALUE_DESCR [DSC$W_LENGTH], .VALUE_DESCR [DSC$A_POINTER]);
;  877          1044  3                     END;
;  878          1045  3
;  879          1046  3                 [INRANGE, OUTRANGE] :
;  880          1047  3                     BAS$$STOP (BAS$K_DATTYPERR);
;  881          1048  3
;  882          1049  3                 TES;
;  883          1050  3
;  884          1051  2             END;
;  885          1052  2
;  886          1053  1     END;                                            ! end of BAS$STORE_BFA
```

```
                                 0FFC 00000          .ENTRY  BAS$STORE_BFA, Save R2,R3,R4,R5,R6,R7,R8,-      ; 0714
                                                             R9,R10,R11
                          5E       24  C2 00002      SUBL2   #36, SP                                         ;
                          50       6C  9A 00005      MOVZBL  (AP), R0                                        ; 0777
                          50       02  C2 00008      SUBL2   #2, R0                                          ;
                          56    08 AC  D0 0000B      MOVL    DESCRIP, R6                                     ;
                          52    0B A6  9A 0000F      MOVZBL  11(R6), R2                                      ;
                          52       50  D1 00013      CMPL    R0, R2                                          ;
                                   1C  13 00016      BEQL    3$                                              ;
                          50       6C  9A 00018      MOVZBL  (AP), R0                                        ; 0781
                          50       02  C2 0001B      SUBL2   #2, R0                                          ;
                          52       50  D1 0001E      CMPL    R0, R2                                          ;
                                   06  1E 00021      BGEQU   1$                                              ;
                          7E    00G 8F  9A 00023      MOVZBL  #BAS$K_TOOFEWARG, -(SP)                         ; 0783
                                   04  11 00027      BRB     2$                                              ;
                          7E    00G 8F  9A 00029 1$:  MOVZBL  #BAS$K_TOOMANARG, -(SP)                         ; 0785
                00000000G 00          01  FB 0002D 2$:  CALLS   #1, BAS$$STOP                                 ;
                05     0A A6          06  E1 00034 3$:  BBC     #6, 10(R6), 4$                                 ; 0793
                               0A A6  95 00039          TSTB    10(R6)                                          ;
                                   0B  19 0003C          BLSS    5$                                              ;
                          7E    00G 8F  9A 0003E 4$:  MOVZBL  #BAS$K_ARGDONMAT, -(SP)                         ;
                00000000G 00          01  FB 00042      CALLS   #1, BAS$$STOP                                   ;
                          55    14 A6  9E 00049 5$:  MOVAB   20(R6), MULTIPLIERS                             ; 0795
                          57    14 A642  DE 0004D      MOVAL   20(R6)[R2], BOUNDS                              ; 0796
                08     0A A6          05  E1 00052      BBC     #5, 10(R6), 6$                                  ; 0802
                          51       52  D0 00057      MOVL    R2, LOW_INDEX                                   ; 0805
                          50       01  D0 0005A      MOVL    #1, HIGH_INDEX                                  ; 0806
                          59       01  CE 0005D      MNEGL   #1, INDEX_INCR                                  ; 0807
                                   09  11 00060      BRB     7$                                              ; 0802
                          51       01  D0 00062 6$:  MOVL    #1, LOW_INDEX                                   ; 0811
                          50       52  D0 00065      MOVL    R2, HIGH_INDEX                                  ; 0812
                          59       01  D0 00068      MOVL    #1, INDEX_INCR                                  ; 0813
                       52    51       59  C3 0006B 7$:  SUBL3   INDEX_INCR, LOW_INDEX, INDEX_NUMBER          ; 0816
                    04 AE    02    A6  9A 0006F      MOVZBL  2(R6), 4(SP)                                    ; 0824
                    1C       00    04  AE 8F 00074      CASEB   4(SP), #0, #28
                003A    003A    003A    0066    00079 8$:  .WORD   12$-8$,-
                003A    003A    003A    003A    00081      9$-8$,-
                003A    003A    003A    003A    00089      9$-8$,-
                003A    003A    003A    003A    00091      9$-8$,-
                003A    003A    003A    003A    00099      9$-8$,-
                003A    003A    0042    003A    000A1      9$-8$,-
                003A    003A    003A    003A    000A9      9$-8$,-
                                        003A    000B1      9$-8$,-
                                                           9$-8$,-
                                                           9$-8$,-
                                                           9$-8$,-
                                                           9$-8$,-
                                                           9$-8$,-
                                                           9$-8$,-
                                                           9$-8$,-
                                                           9$-8$,-
                                                           9$-8$,-
                                                           9$-8$,-
                                                           9$-8$,-
                                                           9$-8$,-
                                                           9$-8$,-
                                                           9$-8$,-
                                                           9$-8$,-
                                                           9$-8$,-
```

```
                                                                 9$-8$,-                                   ;
                                                                 9$-8$,-                                   ;
                                                                 10$-8$,-                                  ;
                                                                 9$-8$,-                                   ;
                                                                 9$-8$,-                                   ;
                                                                 9$-8$,-                                   ;
                                                                 9$-8$,-                                   ;
                                                                 9$-8$,-                                   ;
                                                                 9$-8$,-                                   ;
                                                                 9$-8$                                     ;
               5A                66  3C 000B3 9$:      MOVZWL    (R6), R10                                 ; 0854
               54                5A  D0 000B6          MOVL      R10, LENGTH                               ;
                                 49  11 000B9          BRB       17$                                       ;
               5A                66  3C 000BB 10$:     MOVZWL    (R6), R10                                 ; 0829
 51            5A                02  C7 000BE          DIVL3     #2, R10, R1                               ;
               54          01    A1  9E 000C2          MOVAB     1(R1), LENGTH                             ;
          BF   8F          03    A6  91 000C6          CMPB      3(R6), #191                               ; 0830
                                 37  12 000CB          BNEQ      17$                                       ;
               53                01  D0 000CD          MOVL      #1, I                                     ; 0835
 51            01                53  78 000D0 11$:     ASHL      I, #1, R1                                 ;
               51                54  D1 000D4          CMPL      LENGTH, R1                                ;
                                 1F  19 000D7          BLSS      14$                                       ;
 F3            53                09  F3 000D9          AOBLEQ    #9, I, 11$                                ;
                                 22  11 000DD          BRB       16$                                       ; 0834
               5A                66  3C 000DF 12$:     MOVZWL    (R6), R10                                 ; 0842
               54                5A  D0 000E2          MOVL      R10, LENGTH                               ;
          BF   8F          03    A6  91 000E5          CMPB      3(R6), #191                               ; 0843
                                 18  12 000EA          BNEQ      17$                                       ;
               53                01  D0 000EC          MOVL      #1, I                                     ; 0848
 51            01                53  78 000EF 13$:     ASHL      I, #1, R1                                 ;
               51                54  D1 000F3          CMPL      LENGTH, R1                                ;
                                 05  18 000F6          BGEQ      15$                                       ;
               54                51  D0 000F8 14$:     MOVL      R1, LENGTH                                ; 0849
                                 07  11 000FB          BRB       17$                                       ;
 EE            53                09  F3 000FD 15$:     AOBLEQ    #9, I, 13$                                ; 0848
               54                01  CE 00101 16$:     MNEGL     #1, LENGTH                                ; 0847
                                 53  D4 00104 17$:     CLRL      VALUE_LOCATION                            ; 0860
 5B            50                59  C1 00106          ADDL3     INDEX_INCR, HIGH_INDEX, R11               ; 0862
               52                59  C0 0010A 18$:     ADDL2     INDEX_INCR, INDEX_NUMBER                  ;
               5B                52  D1 0010D          CMPL      INDEX_NUMBER, R11                         ;
                                 2E  13 00110          BEQL      21$                                       ;
               58        08 AC42     D0 00112          MOVL      8(AP)[INDEX_NUMBER], INDEX_VALUE          ; 0864
 50            52                01  78 00117          ASHL      #1, INDEX_NUMBER, R0                      ; 0866
        F8 A740                  58  D1 0011B          CMPL      INDEX_VALUE, -8(BOUNDS)[R0]               ;
                                 07  19 00120          BLSS      19$                                       ;
        FC A740                  58  D1 00122          CMPL      INDEX_VALUE, -4(BOUNDS)[R0]               ; 0867
                                 0B  15 00127          BLEQ      20$                                       ;
               7E         00G    8F  9A 00129 19$:     MOVZBL    #BAS$K_SUBOUTRAN, -(SP)                   ; 0869
     00000000G 00                01  FB 0012D          CALLS     #1, BAS$$STOP                             ;
 50            53        FC A542     C5 00134 20$:     MULL3     -4(MULTIPLIERS)[INDEX_NUMBER], -          ; 0871
                                                                 VALUE_LOCATION, R0                        ;
 53            50                58  C1 0013A          ADDL3     INDEX_VALUE, R0, VALUE_LOCATION           ;
                                 CA  11 0013E          BRB       18$                                       ; 0862
 50            53                54  C5 00140 21$:     MULL3     LENGTH, VALUE_LOCATION, R0                ; 0874
 53            50          10    A6  C1 00144          ADDL3     16(R6), R0, VALUE_LOCATION                ;
                                 52  D4 00149          CLRL      R2                                        ; 0881
               18          04    AE  91 0014B          CMPB      4(SP), #24                                ;
```

```
                                    33  12 0014F          BNEQ    24$                                                     ;
                                    52  D6 00151          INCL    R2                                                      ;
                 18   AE            63  B0 00153          MOVW    (VALUE_LOCATION), VALUE_DESCR                           ; 0888
                 1A   AE       02   A3  90 00157          MOVB    2(VALUE_LOCATION), VALUE_DESCR+2                        ; 0889
                      02       03   A3  91 0015C          CMPB    3(VALUE_LOCATION), #2                                   ; 0890
                                    05  12 00160          BNEQ    22$                                                     ;
                      50            01  D0 00162          MOVL    #1, R0                                                  ;
                                    04  11 00165          BRB     23$                                                     ;
                      50       03   A3  9A 00167 22$:     MOVZBL  3(VALUE_LOCATION), R0                                   ; 0891
                 1B   AE            50  90 0016B 23$:     MOVB    R0, VALUE_DESCR+3                                       ; 0890
                 1C   AE       04   A3  D0 0016F          MOVL    4(VALUE_LOCATION), VALUE_DESCR+4                        ; 0892
                      55            53  D0 00174          MOVL    VALUE_LOCATION, VALUE_DESC_ADDR                         ; 0893
                      15       1A   AE  91 00177          CMPB    VALUE_DESCR+2, #21                                      ; 0894
                                    27  12 0017B          BNEQ    25$                                                     ;
                 20   AE       08   A3  90 0017D          MOVB    8(VALUE_LOCATION), VALUE_DESCR+8                        ; 0899
                                    20  11 00182          BRB     25$                                                     ; 0881
                 1B   AE            5A  B0 00184 24$:     MOVW    R10, VALUE_DESCR                                        ; 0904
                 1A   AE       04   AE  90 00188          MOVB    4(SP), VALUE_DESCR+2                                    ; 0905
                 1B   AE            01  90 0018D          MOVB    #1, VALUE_DESCR+3                                       ; 0906
                 1C   AE            53  D0 00191          MOVL    VALUE_LOCATION, VALUE_DESCR+4                           ; 0907
                      55       18   AE  9E 00195          MOVAB   VALUE_DESCR, VALUE_DESC_ADDR                            ; 0908
                      15       1A   AE  91 00199          CMPB    VALUE_DESCR+2, #21                                      ; 0909
                                    05  12 0019D          BNEQ    25$                                                     ;
                 20   AE       08   A6  90 0019F          MOVB    8(R6), VALUE_DESCR+8                                    ; 0914
                      57       04   AC  D0 001A4 25$:     MOVL    VALUE, R7                                               ; 0923
                 02   A7       1A   AE  91 001A8          CMPB    VALUE_DESCR+2, 2(R7)                                    ;
                                    0B  13 001AD          BEQL    26$                                                     ;
                      7E       00G  8F  9A 001AF          MOVZBL  #BAS$K_DATTYPERR, -(SP)                                 ;
           00000000G  00            01  FB 001B3          CALLS   #1, BAS$$STOP                                           ;
                 BF   8F       03   A6  91 001BA 26$:     CMPB    3(R6), #191                                             ; 0929
                                    03  13 001BF          BEQL    27$                                                     ;
                                00A3  31 001C1          BRW     33$                                                     ;
                      0B            52  E9 001C4 27$:     BLBC    R2, 28$                                                 ; 0933
                      7E       00G  8F  9A 001C7          MOVZBL  #BAS$K_NOTIMP, -(SP)                                    ;
           00000000G  00            01  FB 001CB          CALLS   #1, BAS$$STOP                                           ;
                      0E       04   AE  91 001D2 28$:     CMPB    4(SP), #14                                              ; 0940
                                    76  12 001D6          BNEQ    30$                                                     ;
                 10   AE  020E0000    8F  D0 001D8          MOVL    #34471936, TEMP_DESC                                    ; 0951
                                14   AE  D4 001E0          CLRL    TEMP_DESC+4                                             ; 0954
                                    57  DD 001E3          PUSHL   R7                                                      ; 0955
                                14   AE  9F 001E5          PUSHAB  TEMP_DESC                                               ;
           00000000G  00            02  FB 001E8          CALLS   #2, STR$COPY_DX                                         ;
                      50       10   AE  3C 001EF          MOVZWL  TEMP_DESC, NULLS_COUNT                                  ; 0959
                50    5A            50  C3 001F3          SUBL3   NULLS_COUNT, R10, NULLS_COUNT                           ;
                                    3C  15 001F7          BLEQ    29$                                                     ; 0961
                 08   AE  020E0001    8F  D0 001F9          MOVL    #34471937, NULLS_DESC                                   ; 0968
                                0C   AE  D4 00201          CLRL    NULLS_DESC+4                                            ; 0971
                                04   AE  D4 00204          CLRL    4(SP)                                                   ; 0972
                                04   AE  9F 00207          PUSHAB  4(SP)                                                   ;
                 04   AE            50  D0 0020A          MOVL    NULLS_COUNT, 4(SP)                                      ;
                                04   AE  9F 0020E          PUSHAB  4(SP)                                                   ;
                                10   AE  9F 00211          PUSHAB  NULLS_DESC                                              ;
           00000000G  00            03  FB 00214          CALLS   #3, STR$DUPL_CHAR                                       ;
                                08   AE  9F 00218          PUSHAB  NULLS_DESC                                              ; 0973
                                14   AE  9F 0021E          PUSHAB  TEMP_DESC                                               ;
                                18   AE  9F 00221          PUSHAB  TEMP_DESC                                               ;
           00000000G  00            03  FB 00224          CALLS   #3, STR$CONCAT                                          ;
```

```
                                    08 AE 9F 0022B          PUSHAB  NULLS_DESC                                    ;0974
                00000000G  00       01 FB 0022E             CALLS   #1, STR$FREE1_DX
                                 14 AE DD 00235 29$:        PUSHL   TEMP_DESC+4                                 ;0980
                                    53 DD 00238             PUSHL   VALUE_LOCATION
                                    56 DD 0023A             PUSHL   R6
                00000000G  00       03 FB 0023C             CALLS   #3, BAS$$VA_STORE
                                 10 AE 9F 00243             PUSHAB  TEMP_DESC                                   ;0984
                00000000G  00       01 FB 00246             CALLS   #1, STR$FREE1_DX
                                    04 0024D                RET                                                 ;0940
                           15    04 AE 91 0024E 30$:        CMPB    4(SP), #21                                  ;0988
                                    05 13 00252             BEQL    31$
                                 04 A7 DD 00254             PUSHL   4(R7)                                       ;0990
                                    02 11 00257             BRB     32$
                                    57 DD 00259 31$:        PUSHL   R7                                          ;0992
                                    53 DD 0025B 32$:        PUSHL   VALUE_LOCATION
                                    56 DD 0025D             PUSHL   R6
                00000000G  00       03 FB 0025F             CALLS   #3, BAS$$VA_STORE
                                    04 00266                RET                                                 ;0940
                           04    03 A6 91 00267 33$:        CMPB    3(R6), #4                                   ;0998
                                    0B 13 0026B             BEQL    34$
                           7E    00G 8F 9A 0026D           MOVZBL  #BAS$K_NOTIMP, -(SP)
                00000000G  00       01 FB 00271             CALLS   #1, BAS$$STOP
                        1C  00    02 A7 8F 00278 34$:       CASEB   2(R7), #0, #28                              ;1005
                003A    003A    003A   0094    00279 35$:   .WORD    44$-35$,-
                004C    0046    003A   003A    00285        .WORD    36$-35$,-
                0067    0058    003A   0052    0028D        .WORD    36$-35$,-
                003A    0094    003A   003A    00295        .WORD    36$-35$,-
                003A    003A    003A   003A    0029D        .WORD    36$-35$,-
                003A    003A    00A0   003A    002A5        .WORD    36$-35$,-
                0076    003A    003A   003A    002AD        .WORD    37$-35$,-
                                       0085    002B5        .WORD    38$-35$,-
                                                            .WORD    39$-35$,-
                                                            .WORD    36$-35$,-
                                                            .WORD    40$-35$,-
                                                            .WORD    41$-35$,-
                                                            .WORD    36$-35$,-
                                                            .WORD    36$-35$,-
                                                            .WORD    44$-35$,-
                                                            .WORD    36$-35$,-
                                                            .WORD    36$-35$,-
                                                            .WORD    36$-35$,-
                                                            .WORD    36$-35$,-
                                                            .WORD    36$-35$,-
                                                            .WORD    36$-35$,-
                                                            .WORD    45$-35$,-
                                                            .WORD    36$-35$,-
                                                            .WORD    36$-35$,-
                                                            .WORD    36$-35$,-
                                                            .WORD    36$-35$,-
                                                            .WORD    36$-35$,-
                                                            .WORD    36$-35$,-
                                                            .WORD    42$-35$,-
                                                            .WORD    43$-35$
                           7E    00G 8F 9A 002B7 36$:       MOVZBL  #BAS$K_DATTYPERR, -(SP)                     ;1047
                00000000G  00       01 FB 002BB             CALLS   #1, BAS$$STOP
                                    04 002C2                RET
                        1C  BE    04 B7 90 002C3 37$:       MOVB    @4(R7), @VALUE_DESCR+4                      ;1010
                                    04 002C8                RET
```

```
                    1C  BE      04  B7  B0 002C9 38$:  MOVW    @4(R7), @VALUE_DESCR+4                  ; 1014
                                        04 002CE        RET
                    1C  BE      04  B7  D0 002CF 39$:  MOVL    @4(R7), @VALUE_DESCR+4                  ; 1018
                                        04 002D4        RET
                        51      1C  AE  D0 002D5 40$:  MOVL    VALUE_DESCR+4, R1                       ; 1021
                        50      04  A7  D0 002D9       MOVL    4(R7), R0
                    00000000G   00      16 002DD       JSB     BAS$$COPY_F_R1
                                        04 002E3       RET
                        51      1C  AE  D0 002E4 41$:  MOVL    VALUE_DESCR+4, R1                       ; 1024
                        50      04  A7  D0 002E8       MOVL    4(R7), R0
                    00000000G   00      16 002EC       JSB     BAS$$COPY_D_R1
                                        04 002F2       RET
                        51      1C  AE  D0 002F3 42$:  MOVL    VALUE_DESCR+4, R1                       ; 1027
                        50      04  A7  D0 002F7       MOVL    4(R7), R0
                    00000000G   00      16 002FB       JSB     BAS$$COPY_G_R1
                                        04 00301       RET
                        51      1C  AE  D0 00302 43$:  MOVL    VALUE_DESCR+4, R1                       ; 1030
                        50      04  A7  D0 00306       MOVL    4(R7), R0
                    00000000G   00      16 0030A       JSB     BAS$$COPY_H_R3
                                        04 00310       RET
                            00A0    8F  BB 00311 44$:  PUSHR   #^M<R5,R7>                              ; 1033
                00000000G   00          02  FB 00315       CALLS   #2, STR$COPY_DX
                                        04 0031C       RET
                        50      08  A7  98 0031D 45$:  CVTBL   8(R7), COUNT                           ; 1041
                        51      20  AE  98 00321       CVTBL   VALUE_DESCR+8, R1
                        50          51  C2 00325       SUBL2   R1, COUNT
        00      04  B7              67  50  FB 00328       ASHP    COUNT, (R7), @4(R7), #0, VALUE_DESCR, -     ; 1043
                    1C  BE      18  AE          0032E               @VALUE_DESCR+4
                                        04 00332       RET                                             ; 1053

; Routine Size:  819 bytes,    Routine Base:  _BAS$CODE + 0312

;  887          1054  1
```

```
;  889          1055  1 GLOBAL ROUTINE BAS$STORE_BFA_OFF (                    ! Store a value by desc w/offset
;  890          1056  1         VALUE : REF BLOCK [8, BYTE],                  ! Where to find the value
;  891          1057  1         DESCRIP : REF BLOCK [8, BYTE],                ! The descriptor to store it
;  892          1058  1         OFFSET,                                       ! Offset into record
;  893          1059  1         STR_LENGTH,                                   ! String length (if string VALUE)
;  894          1060  1         INDEX1                                        ! First index
;  895          1061  1     ) : NOVALUE =
;  896          1062  1
;  897          1063  1 !++
;  898          1064  1 ! FUNCTIONAL DESCRIPTION:
;  899          1065  1 !
;  900          1066  1 !       Store a value in an array or virtual array.  The location from
;  901          1067  1 !       which to fetch the value is passed by descriptor.  This is a special
;  902          1068  1 !       routine to handle record arrays - the offset specifies an element
;  903          1069  1 !       in the record.
;  904          1070  1 !
;  905          1071  1 ! FORMAL PARAMETERS:
;  906          1072  1 !
;  907          1073  1 !       VALUE.rx.dx     The place from which to get the value stored
;  908          1074  1 !       DESCRIP.rx.da   The descriptor of the array or virtual array
;  909          1075  1 !       OFFSET.rlu.v    Offset into the record
;  910          1076  1 !       STR_LENGTH.rlu.v Length if string VALUE, 0 otherwise
;  911          1077  1 !       INDEX1.rl.v     The first index into the array.  More indicies
;  912          1078  1 !                       may follow this one in the calling sequence.
;  913          1079  1 !
;  914          1080  1 ! IMPLICIT INPUTS:
;  915          1081  1 !
;  916          1082  1 !       NONE
;  917          1083  1 !
;  918          1084  1 ! IMPLICIT OUTPUTS:
;  919          1085  1 !
;  920          1086  1 !       NONE
;  921          1087  1 !
;  922          1088  1 ! ROUTINE VALUE:
;  923          1089  1 ! COMPLETION CODES:
;  924          1090  1 !
;  925          1091  1 !       NONE
;  926          1092  1 !
;  927          1093  1 ! SIDE EFFECTS:
;  928          1094  1 !
;  929          1095  1 !       Signals if an error is encountered.
;  930          1096  1 !
;  931          1097  1 !--
;  932          1098  1
;  933          1099  2     BEGIN
;  934          1100  2
;  935          1101  2     BUILTIN
;  936          1102  2         ACTUALCOUNT,
;  937          1103  2         ACTUALPARAMETER,
;  938          1104  2         ASHP;
;  939          1105  2
;  940          1106  2     LOCAL
;  941          1107  2         INDEX_VALUE,
;  942          1108  2         VALUE_LOCATION,
;  943          1109  2         MULTIPLIERS : REF VECTOR,
;  944          1110  2         BOUNDS : REF VECTOR,
;  945          1111  2         LOW_INDEX,
```

```
;  946          1112  2          HIGH_INDEX,
;  947          1113  2          INDEX_INCR,
;  948          1114  2          INDEX_NUMBER,
;  949          1115  2          VALUE_DESCR : BLOCK [12, BYTE],
;  950          1116  2          VALUE_DESC_ADDR,
;  951          1117  2          LENGTH;
;  952          1118  2
;  953          1119  2      !+
;  954          1120  2      ! Be sure the number of array subscripts matches the number of
;  955          1121  2      ! indicies given to us.
;  956          1122  2      !-
;  957          1123  2
;  958          1124  2          IF ((ACTUALCOUNT () - 4) NEQU .DESCRIP [DSC$B_DIMCT])
;  959          1125  2          THEN
;  960          1126  2              BEGIN
;  961          1127  2
;  962          1128  4              IF ((ACTUALCOUNT () - 4) LSSU .DESCRIP [DSC$B_DIMCT])
;  963          1129  4              THEN
;  964          1130  4                  BAS$$STOP (BAS$K_TOOFEWARG)
;  965          1131  4              ELSE
;  966          1132  4                  BAS$$STOP (BAS$K_TOOMANARG);
;  967          1133  4
;  968          1134  4              END;
;  969          1135  2
;  970          1136  2      !+
;  971          1137  2      ! The coefficients and bounds must be present.
;  972          1138  2      !-
;  973          1139  2
;  974          1140  2          IF ( NOT (.DESCRIP [DSC$V_FL_COEFF] AND .DESCRIP [DSC$V_FL_BOUNDS])) THEN BAS$$STOP (BAS$K_ARGDONMAT);
;  975          1141  2
;  976          1142  2          MULTIPLIERS = DESCRIP [DSC$L_M1];
;  977          1143  2          BOUNDS = DESCRIP [DSC$L_M1] + (%UPVAL*.DESCRIP [DSC$B_DIMCT]);
;  978          1144  2      !+
;  979          1145  2      ! Compute the lower and upper index numbers based on how the array
;  980          1146  2      ! is stored.
;  981          1147  2      !-
;  982          1148  2
;  983          1149  3          IF (.DESCRIP [DSC$V_FL_COLUMN])
;  984          1150  2          THEN
;  985          1151  3              BEGIN
;  986          1152  3              LOW_INDEX = .DESCRIP [DSC$B_DIMCT];
;  987          1153  3              HIGH_INDEX = 1;
;  988          1154  3              INDEX_INCR = -1;
;  989          1155  3              END
;  990          1156  2          ELSE
;  991          1157  3              BEGIN
;  992          1158  3              LOW_INDEX = 1;
;  993          1159  3              HIGH_INDEX = .DESCRIP [DSC$B_DIMCT];
;  994          1160  3              INDEX_INCR = 1;
;  995          1161  3              END;
;  996          1162  2
;  997          1163  2          INDEX_NUMBER = .LOW_INDEX - .INDEX_INCR;
;  998          1164  2      !+
;  999          1165  2      ! If this is a decimal array, the length in the descriptor is the number of
; 1000          1166  2      ! 4 bit digits (not including the sign).  Convert this length to the number
; 1001          1167  2      ! of bytes.
; 1002          1168  2      ! Also, if this is a virtual array, the size must be a multiple of 2.  This
```

```
;1003     1169  2 ! is true for arrays of records as well.
;1004     1170  2 !-
;1005     1171  2     CASE .DESCRIP [DSC$B_DTYPE] FROM DSC$K_DTYPE_Z TO DSC$K_DTYPE_H OF
;1006     1172  2         SET
;1007     1173  2
;1008     1174  2         [DSC$K_DTYPE_P] :                   ! decimal
;1009     1175  3             BEGIN
;1010     1176  3             LENGTH = (.DESCRIP [DSC$W_LENGTH]/2) + 1;
;1011     1177  3             IF .DESCRIP [DSC$B_CLASS] EQL DSC$K_CLASS_BFA
;1012     1178  3             THEN
;1013     1179  4                 BEGIN
;1014     1180  4
;1015     1181  5                 LENGTH = ( INCR I FROM 1 TO 9 BY 1 DO
;1016     1182  6                             IF .LENGTH LSS (1 ^ .I)
;1017     1183  4                             THEN EXITLOOP (1 ^ .I) );
;1018     1184  3                 END;
;1019     1185  2             END;
;1020     1186  2
;1021     1187  2         [DSC$K_DTYPE_Z] :                   ! record
;1022     1188  3             BEGIN
;1023     1189  3             LENGTH = .DESCRIP [DSC$W_LENGTH];
;1024     1190  3             IF .DESCRIP [DSC$B_CLASS] EQL DSC$K_CLASS_BFA
;1025     1191  3             THEN
;1026     1192  4                 BEGIN
;1027     1193  4
;1028     1194  5                 LENGTH = ( INCR I FROM 1 TO 9 BY 1 DO
;1029     1195  6                             IF .LENGTH LSS (1 ^ .I)
;1030     1196  4                             THEN EXITLOOP (1 ^ .I) );
;1031     1197  3                 END;
;1032     1198  2             END;
;1033     1199  2
;1034     1200  2         [INRANGE, OUTRANGE] :
;1035     1201  2             LENGTH = .DESCRIP [DSC$W_LENGTH];
;1036     1202  2         TES;
;1037     1203  2
;1038     1204  2 !+
;1039     1205  2 ! Compute the linear index from the indices provided.
;1040     1206  2 !-
;1041     1207  2     VALUE_LOCATION = 0;
;1042     1208  2
;1043     1209  2     WHILE ((INDEX_NUMBER = .INDEX_NUMBER + .INDEX_INCR) NEQ (.HIGH_INDEX + .INDEX_INCR)) DO
;1044     1210  3         BEGIN
;1045     1211  3         INDEX_VALUE = ACTUALPARAMETER (.INDEX_NUMBER + 4);
;1046     1212  3
;1047     1213  3         IF ((.INDEX_VALUE LSS .BOUNDS [(.INDEX_NUMBER - 1)*2])  !
;1048     1214  4             OR (.INDEX_VALUE GTR .BOUNDS [((.INDEX_NUMBER - 1)*2) + 1]))
;1049     1215  3         THEN
;1050     1216  3             BAS$$STOP (BAS$K_SUBOUTRAN);
;1051     1217  3
;1052     1218  3         VALUE_LOCATION = (.VALUE_LOCATION*.MULTIPLIERS [.INDEX_NUMBER - 1]) + .INDEX_VALUE;
;1053     1219  2         END;
;1054     1220  2
;1055     1221  2     VALUE_LOCATION = (.VALUE_LOCATION*.LENGTH) + .DESCRIP [DSC$A_A0];
;1056     1222  2
;1057     1223  2 !+
;1058     1224  2 ! Add the offset to the linear index to arrive at the desired element within
;1059     1225  2 ! the record.
```

```
;1060     1226  2 !-
;1061     1227  2     VALUE_LOCATION = .VALUE_LOCATION + .OFFSET;
;1062     1228  2
;1063     1229  2 !+
;1064     1230  2 ! Build a descriptor pointing to the value cell in the array.  If this
;1065     1231  2 ! is an array of descriptors, the descriptor is copied, otherwise it
;1066     1232  2 ! is constructed.
;1067     1233  2 !-
;1068     1234  2
;1069     1235  3     IF (.DESCRIP [DSC$B_DTYPE] EQLU DSC$K_DTYPE_DSC)
;1070     1236  3     THEN
;1071     1237  3         BEGIN
;1072     1238  3
;1073     1239  3         MAP
;1074     1240  3             VALUE_LOCATION : REF BLOCK [8, BYTE];
;1075     1241  3
;1076     1242  3         VALUE_DESCR [DSC$W_LENGTH] = .VALUE_LOCATION [DSC$W_LENGTH];
;1077     1243  3         VALUE_DESCR [DSC$B_DTYPE] = .VALUE_LOCATION [DSC$B_DTYPE];
;1078     1244  4         VALUE_DESCR [DSC$B_CLASS] = (IF (.VALUE_LOCATION [DSC$B_CLASS] EQLU DSC$K_CLASS_D) THEN DSC$K_CLASS_
;1079     1245  3             ELSE .VALUE_LOCATION [DSC$B_CLASS]);
;1080     1246  3         VALUE_DESCR [DSC$A_POINTER] = .VALUE_LOCATION [DSC$A_POINTER];
;1081     1247  3         VALUE_DESC_ADDR = .VALUE_LOCATION;
;1082     1248  3         IF .VALUE_DESCR [DSC$B_DTYPE] EQL DSC$K_DTYPE_P
;1083     1249  3         THEN
;1084     1250  4             BEGIN
;1085     1251  4             MAP
;1086     1252  4                 VALUE_LOCATION : REF BLOCK [12,BYTE];
;1087     1253  4             VALUE_DESCR [DSC$B_SCALE] = .VALUE_LOCATION [DSC$B_SCALE];
;1088     1254  3             END;
;1089     1255  3         END
;1090     1256  2     ELSE
;1091     1257  3         BEGIN
;1092     1258  4         VALUE_DESCR [DSC$W_LENGTH] = (IF .VALUE [DSC$B_DTYPE] EQL DSC$K_DTYPE_T
;1093     1259  4                                       OR .VALUE [DSC$B_DTYPE] EQL DSC$K_DTYPE_Z
;1094     1260  4                                       THEN .STR_LENGTH
;1095     1261  3                                       ELSE .VALUE [DSC$W_LENGTH]);
;1096     1262  3         VALUE_DESCR [DSC$B_DTYPE] = .VALUE [DSC$B_DTYPE];
;1097     1263  3         VALUE_DESCR [DSC$B_CLASS] = DSC$K_CLASS_S;
;1098     1264  3         VALUE_DESCR [DSC$A_POINTER] = .VALUE_LOCATION;
;1099     1265  3         VALUE_DESC_ADDR = VALUE_DESCR;
;1100     1266  3         IF .VALUE_DESCR [DSC$B_DTYPE] EQL DSC$K_DTYPE_P
;1101     1267  3         THEN
;1102     1268  4             BEGIN
;1103     1269  4             MAP
;1104     1270  4                 DESCRIP : REF BLOCK [12,BYTE];
;1105     1271  4             VALUE_DESCR [DSC$B_SCALE] = .DESCRIP [DSC$B_SCALE];
;1106     1272  3             END;
;1107     1273  2         END;
;1108     1274  2
;1109     1275  2 !+
;1110     1276  2 ! Make sure that the data type of the array element agrees with
;1111     1277  2 ! that of the source descriptor.
;1112     1278  2 !-
;1113     1279  2
;1114     1280  2     IF (.VALUE_DESCR [DSC$B_DTYPE] NEQU .VALUE [DSC$B_DTYPE]) THEN BAS$$STOP (BAS$K_DATTYPERR);
;1115     1281  2
;1116     1282  2 !+
```

```
;  1117          1283  2 ! Special handling if this is a virtual array.
;  1118          1284  2 !-
;  1119          1285  2
;  1120          1286  2     IF (.DESCRIP [DSC$B_CLASS] EQLU DSC$K_CLASS_BFA)
;  1121          1287  2     THEN
;  1122          1288  3         BEGIN
;  1123          1289  3
;  1124          1290  3         LOCAL
;  1125          1291  3             SAVE_LENGTH;
;  1126          1292  3         !+
;  1127          1293  3         ! KLUDGE!!! dummy up the array length descriptor to contain
;  1128          1294  3         !           the length of the element in the record we are
;  1129          1295  3         !           interested in.
;  1130          1296  3         !-
;  1131          1297  3         SAVE_LENGTH = .DESCRIP [DSC$W_LENGTH];
;  1132          1298  3         DESCRIP [DSC$W_LENGTH] = ( IF DESCRIP [DSC$B_DTYPE] EQL DSC$K_DTYPE_P
;  1133          1299  3                                    THEN .VALUE_DESCR [DSC$W_LENGTH] / 2 + 1
;  1134          1300  3                                    ELSE .VALUE_DESCR [DSC$W_LENGTH] );
;  1135          1301  3
;  1136          1302  3         IF (.DESCRIP [DSC$B_DTYPE] EQLU DSC$K_DTYPE_DSC) THEN BAS$$STOP (BAS$K_NOTIMP);
;  1137          1303  3
;  1138          1304  3     !+
;  1139          1305  3     ! If this is a string, we must pad it with SPACES.  To do this, we need
;  1140          1306  3     ! a temporary string.
;  1141          1307  3     !-
;  1142          1308  3
;  1143          1309  4         IF (.VALUE_DESCR [DSC$B_DTYPE] EQL DSC$K_DTYPE_T)
;  1144          1310  4         OR (.VALUE_DESCR [DSC$B_DTYPE] EQL DSC$K_DTYPE_Z)
;  1145          1311  4         THEN
;  1146          1312  4             BEGIN
;  1147          1313  4
;  1148          1314  4             LOCAL
;  1149          1315  4                 SPACES_COUNT,
;  1150          1316  4                 TEMP_DESC : BLOCK [8, BYTE];
;  1151          1317  4
;  1152          1318  4             !+
;  1153          1319  4             ! Copy the caller's string to our temporary.
;  1154          1320  4             !-
;  1155          1321  4             TEMP_DESC [DSC$W_LENGTH] = 0;
;  1156          1322  4             TEMP_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  1157          1323  4             TEMP_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  1158          1324  4             TEMP_DESC [DSC$A_POINTER] = 0;
;  1159          1325  4             STR$COPY_DX (TEMP_DESC, .VALUE);
;  1160          1326  4             !+
;  1161          1327  4             ! Concatenate enough SPACES onto the string to make it the right length.
;  1162          1328  4             !-
;  1163          1329  4             SPACES_COUNT = .VALUE_DESCR [DSC$W_LENGTH] - .TEMP_DESC [DSC$W_LENGTH];
;  1164          1330  4
;  1165          1331  5             IF (.SPACES_COUNT GTR 0)
;  1166          1332  4             THEN
;  1167          1333  5                 BEGIN
;  1168          1334  5
;  1169          1335  5                 LOCAL
;  1170          1336  5                     SPACES_DESC : BLOCK [8, BYTE];
;  1171          1337  5
;  1172          1338  5                 SPACES_DESC [DSC$W_LENGTH] = 1;
;  1173          1339  5                 SPACES_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
```

```
;  1174          1340  5                  SPACES_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  1175          1341  5                  SPACES_DESC [DSC$A_POINTER] = 0;
;  1176          1342  5                  STR$DUPL_CHAR (SPACES_DESC, %REF (.SPACES_COUNT), %REF (%X'20'));
;  1177          1343  5                  STR$CONCAT (TEMP_DESC, TEMP_DESC, SPACES_DESC);
;  1178          1344  5                  STR$FREE1_DX (SPACES_DESC);
;  1179          1345  4                  END;
;  1180          1346  4
;  1181          1347  4          !+
;  1182          1348  4          ! Now store the (possibly lengthened) string in the file.
;  1183          1349  4          !-
;  1184          1350  4                  BAS$$VA_STORE (.DESCRIP, .VALUE_LOCATION, .TEMP_DESC [DSC$A_POINTER]);
;  1185          1351  4          !+
;  1186          1352  4          ! Free our temporary string.
;  1187          1353  4          !-
;  1188          1354  4                  STR$FREE1_DX (TEMP_DESC);
;  1189          1355  4                  END
;  1190          1356  3              ELSE
;  1191          1357  4                  BEGIN
;  1192          1358  4                      BAS$$VA_STORE (.DESCRIP, .VALUE_LOCATION, .VALUE [DSC$A_POINTER])
;  1193          1359  3                  END;
;  1194          1360  3
;  1195          1361  3                  !+
;  1196          1362  3                  ! put length field in array descriptor back the way
;  1197          1363  3                  ! it was before we KLUDGED it!
;  1198          1364  3                  !-
;  1199          1365  3                  DESCRIP [DSC$W_LENGTH] = .SAVE_LENGTH;
;  1200          1366  3                  END
;  1201          1367  2          ELSE
;  1202          1368  3              BEGIN
;  1203          1369  3
;  1204          1370  3              IF (.DESCRIP [DSC$B_CLASS] NEQU DSC$K_CLASS_A) THEN BAS$$STOP (BAS$K_NOTIMP);
;  1205          1371  3
;  1206          1372  3          !+
;  1207          1373  3          ! Copy the value cell to the array element.  The form of the copy is
;  1208          1374  3          ! based on the type of data.
;  1209          1375  3          !-
;  1210          1376  3
;  1211          1377  3              CASE .VALUE [DSC$B_DTYPE] FROM DSC$K_DTYPE_Z TO DSC$K_DTYPE_H OF
;  1212          1378  3                  SET
;  1213          1379  3
;  1214          1380  3                  [DSC$K_DTYPE_B] :                   ! byte
;  1215          1381  3                      BLOCK [.VALUE_DESCR [DSC$A_POINTER], 0, 0, %BPUNIT, 1]
;  1216          1382  3                      = .BLOCK [.VALUE [DSC$A_POINTER], 0, 0, %BPUNIT, 1];
;  1217          1383  3
;  1218          1384  3                  [DSC$K_DTYPE_W] :                   ! 16-bit word
;  1219          1385  3                      BLOCK [.VALUE_DESCR [DSC$A_POINTER], 0, 0, %BPVAL/2, 1] !
;  1220          1386  3                      = .BLOCK [.VALUE [DSC$A_POINTER], 0, 0, %BPVAL/2, 1];
;  1221          1387  3
;  1222          1388  3                  [DSC$K_DTYPE_L] :                   ! 32-bit longword
;  1223          1389  3                      BLOCK [.VALUE_DESCR [DSC$A_POINTER], 0, 0, %BPVAL, 1]   !
;  1224          1390  3                      = .BLOCK [.VALUE [DSC$A_POINTER], 0, 0, %BPVAL, 1];
;  1225          1391  3
;  1226          1392  3                  [DSC$K_DTYPE_F] :                   ! 32-bit floating point
;  1227          1393  3                      BAS$$COPY_F_R1 (.VALUE [DSC$A_POINTER], .VALUE_DESCR [DSC$A_POINTER]);
;  1228          1394  3
;  1229          1395  3                  [DSC$K_DTYPE_D] :                   ! 64-bit double floating
;  1230          1396  3                      BAS$$COPY_D_R1 (.VALUE [DSC$A_POINTER], .VALUE_DESCR [DSC$A_POINTER]);
```

```
;  1231          1397  3              [DSC$K_DTYPE_G] :                       ! G floating
;  1232          1398  3                  BAS$$COPY_G_R1 (.VALUE [DSC$A_POINTER], .VALUE_DESCR [DSC$A_POINTER]);
;  1233          1399  3
;  1234          1400  3              [DSC$K_DTYPE_H] :                       ! H floating
;  1235          1401  3                  BAS$$COPY_H_R3 (.VALUE [DSC$A_POINTER], .VALUE_DESCR [DSC$A_POINTER]);
;  1236          1402  3
;  1237          1403  3              [DSC$K_DTYPE_T, DSC$K_DTYPE_Z] :        ! Text string or record
;  1238          1404  3                  STR$COPY_DX (.VALUE_DESC_ADDR, .VALUE);
;  1239          1405  3
;  1240          1406  3              [DSC$K_DTYPE_P] :                       ! decimal
;  1241          1407  3                  BEGIN
;  1242          1408  4                  MAP
;  1243          1409  4                      VALUE : REF BLOCK [12,BYTE];
;  1244          1410  4                  LOCAL
;  1245          1411  4                      COUNT;
;  1246          1412  4                  COUNT = .VALUE [DSC$B_SCALE] - .VALUE_DESCR [DSC$B_SCALE];
;  1247          1413  4                  ASHP (COUNT, VALUE [DSC$W_LENGTH], .VALUE [DSC$A_POINTER],
;  1248          1414  4                      %REF(0), VALUE_DESCR [DSC$W_LENGTH], .VALUE_DESCR [DSC$A_POINTER]);
;  1249          1415  4                  END;
;  1250          1416  3
;  1251          1417  3              [INRANGE, OUTRANGE] :
;  1252          1418  3                  BAS$$STOP (BAS$K_DATTYPERR);
;  1253          1419  3
;  1254          1420  3              TES;
;  1255          1421  3
;  1256          1422  2          END;
;  1257          1423  2
;  1258          1424  1      END;                                            ! end of BAS$STORE_BFA_OFF
;  1259          1425  1
```

```
                                 0FFC 00000          .ENTRY  BAS$STORE_BFA_OFF, Save R2,R3,R4,R5,R6,R7,-      ; 1055
                                                             R8,R9,R10,R11
                          5E          24  C2 00002           SUBL2   #36, SP
                          50          6C  9A 00005           MOVZBL  (AP), R0                                 ; 1124
                          50          04  C2 00008           SUBL2   #4, R0
                          56    0B    AC  D0 0000B           MOVL    DESCRIP, R6
                          52    0B    A6  9A 0000F           MOVZBL  11(R6), R2
                          52          50  D1 00013           CMPL    R0, R2
                                      1C  13 00016           BEQL    3$
                          50          6C  9A 00018           MOVZBL  (AP), R0                                 ; 1128
                          50          04  C2 0001B           SUBL2   #4, R0
                          52          50  D1 0001E           CMPL    R0, R2
                                      06  1E 00021           BGEQU   1$
                          7E    00G   8F  9A 00023           MOVZBL  #BAS$K_TOOFEWARG, -(SP)                  ; 1130
                                      04  11 00027           BRB     2$
                          7E    00G   8F  9A 00029 1$:       MOVZBL  #BAS$K_TOOMANARG, -(SP)                  ; 1132
                00000000G 00          01  FB 0002D 2$:       CALLS   #1, BAS$$STOP
                    05      0A    A6      E1 00034 3$:       BBC     #6, 10(R6), 4$                           ; 1140
                                0A    A6  95 00039           TSTB    10(R6)
                                      0B  19 0003C           BLSS    5$
                          7E    00G   8F  9A 0003E 4$:       MOVZBL  #BAS$K_ARGDONMAT, -(SP)
                00000000G 00          01  FB 00042           CALLS   #1, BAS$$STOP
                          55    14    A6  9E 00049 5$:       MOVAB   20(R6), MULTIPLIERS                      ; 1142
```

```
                            57      14 A642  DE 0004D        MOVAL   20(R6)[R2], BOUNDS                          ; 1143
          0B          0A    A6          05  E1 00052        BBC     #5, 10(R6), 6$                              ; 1149
                            51          52  D0 00057        MOVL    R2, LOW_INDEX                               ; 1152
                            50          01  D0 0005A        MOVL    #1, HIGH_INDEX                              ; 1153
                            5A          01  CE 0005D        MNEGL   #1, INDEX_INCR                              ; 1154
                                        09  11 00060        BRB     7$                                          ; 1149
                            51          01  D0 00062 6$:    MOVL    #1, LOW_INDEX                               ; 1158
                            50          52  D0 00065        MOVL    R2, HIGH_INDEX                              ; 1159
                            5A          01  D0 00068        MOVL    #1, INDEX_INCR                              ; 1160
          52                51          5A  C3 0006B 7$:    SUBL3   INDEX_INCR, LOW_INDEX, INDEX_NUMBER         ; 1163
          1C                00      02  A6  8F 0006F        CASEB   2(R6), #0, #28                              ; 1171
 003A     003A            003A        0066     00074 8$:    .WORD   12$-8$,-
 003A     003A            003A        003A     0007C                9$-8$,-
 003A     003A            003A        003A     00084                9$-8$,-
 003A     003A            003A        003A     0008C                9$-8$,-
 003A     003A            003A        003A     00094                9$-8$,-
 003A     003A            0042        003A     0009C                9$-8$,-
 003A     003A            003A        003A     000A4                9$-8$,-
                                      003A     000AC                9$-8$,-
                                                                    9$-8$,-
                                                                    9$-8$,-
                                                                    9$-8$,-
                                                                    9$-8$,-
                                                                    9$-8$,-
                                                                    9$-8$,-
                                                                    9$-8$,-
                                                                    9$-8$,-
                                                                    9$-8$,-
                                                                    9$-8$,-
                                                                    9$-8$,-
                                                                    9$-8$,-
                                                                    9$-8$,-
                                                                    10$-8$,-
                                                                    9$-8$,-
                                                                    9$-8$,-
                                                                    9$-8$,-
                                                                    9$-8$,-
                                                                    9$-8$,-
                                                                    9$-8$,-
                                                                    9$-8$
                            58          66  3C 000AE 9$:    MOVZWL  (R6), R8                                    ; 1201
                            54          58  D0 000B1        MOVL    R8, LENGTH
                                        49  11 000B4        BRB     17$
                            58          66  3C 000B6 10$:   MOVZWL  (R6), R8                                    ; 1176
          51                58          02  C7 000B9        DIVL3   #2, R8, R1
                            54      01  A1  9E 000BD        MOVAB   1(R1), LENGTH
                    BF      8F      03  A6  91 000C1        CMPB    3(R6), #191                                 ; 1177
                                        37  12 000C6        BNEQ    17$
                            51          01  D0 000C8        MOVL    #1, I                                       ; 1182
          53                01          51  78 000CB 11$:   ASHL    I, #1, R3
                            53          54  D1 000CF        CMPL    LENGTH, R3
                                        1F  19 000D2        BLSS    14$
          F3                51          09  F3 000D4        AOBLEQ  #9, I, 11$
                                        22  11 000D8        BRB     16$                                         ; 1181
                            58          66  3C 000DA 12$:   MOVZWL  (R6), R8                                    ; 1189
                            54          58  D0 000DD        MOVL    R8, LENGTH
                    BF      8F      03  A6  91 000E0        CMPB    3(R6), #191                                 ; 1190
```

```
                                    18   12 000E5          BNEQ     17$                                          ;
                    51              01   D0 000E7          MOVL     #1, I                                        ; 1195
53                  01              51   78 000EA  13$:    ASHL     I, #1, R3                                    ;
                    53              54   D1 000EE          CMPL     LENGTH, R3                                   ;
                                    05   18 000F1          BGEQ     15$                                          ;
                    54              53   D0 000F3  14$:    MOVL     R3, LENGTH                                   ; 1196
                                    07   11 000F6          BRB      17$                                          ;
EE                  51              09   F3 000F8  15$:    AOBLEQ   #9, I, 13$                                   ; 1195
                    54              01   CE 000FC  16$:    MNEGL    #1, LENGTH                                   ; 1194
                                    53   D4 000FF  17$:    CLRL     VALUE_LOCATION                               ; 1207
5B                  50              5A   C1 00101          ADDL3    INDEX_INCR, HIGH_INDEX, R11                  ; 1209
                    52              5A   C0 00105  18$:    ADDL2    INDEX_INCR, INDEX_NUMBER                     ;
                    5B              52   D1 00108          CMPL     INDEX_NUMBER, R11                            ;
                                    2E   13 0010B          BEQL     21$                                          ;
                    59         10 AC42   D0 0010D          MOVL     16(AP)[INDEX_NUMBER], INDEX_VALUE            ; 1211
50                  52              01   78 00112          ASHL     #1, INDEX_NUMBER, R0                         ; 1213
              F8 A740               59   D1 00116          CMPL     INDEX_VALUE, -8(BOUNDS)[R0]                  ;
                                    07   19 0011B          BLSS     19$                                          ;
              FC A740               59   D1 0011D          CMPL     INDEX_VALUE, -4(BOUNDS)[R0]                  ; 1214
                                    0B   15 00122          BLEQ     20$                                          ;
                    7E         00G  8F   9A 00124  19$:    MOVZBL   #BAS$K_SUBOUTRAN, -(SP)                      ; 1216
         00000000G  00              01   FB 00128          CALLS    #1, BAS$$STOP                                ;
50                  53         FC A542   C5 0012F  20$:    MULL3    -4(MULTIPLIERS)[INDEX_NUMBER], -             ; 1218
                                                                    VALUE_LOCATION, R0                           ;
53                  50              59   C1 00135          ADDL3    INDEX_VALUE, R0, VALUE_LOCATION              ;
                                    CA   11 00139          BRB      18$                                          ; 1209
50                  53              54   C5 0013B  21$:    MULL3    LENGTH, VALUE_LOCATION, R0                   ; 1221
53                  50         10   A6   C1 0013F          ADDL3    16(R6), R0, VALUE_LOCATION                   ;
                    53         0C   AC   C0 00144          ADDL2    OFFSET, VALUE_LOCATION                       ; 1227
                    18         02   A6   91 00148          CMPB     2(R6), #24                                   ; 1235
                                    31   12 0014C          BNEQ     24$                                          ;
              18    AE              63   B0 0014E          MOVW     (VALUE_LOCATION), VALUE_DESCR                ; 1242
              1A    AE         02   A3   90 00152          MOVB     2(VALUE_LOCATION), VALUE_DESCR+2             ; 1243
                    02         03   A3   91 00157          CMPB     3(VALUE_LOCATION), #2                        ; 1244
                                    05   12 0015B          BNEQ     22$                                          ;
                    50              01   D0 0015D          MOVL     #1, R0                                       ;
                                    04   11 00160          BRB      23$                                          ;
                    50         03   A3   9A 00162  22$:    MOVZBL   3(VALUE_LOCATION), R0                        ; 1245
              1B    AE              50   90 00166  23$:    MOVB     R0, VALUE_DESCR+3                            ; 1244
              1C    AE         04   A3   D0 0016A          MOVL     4(VALUE_LOCATION), VALUE_DESCR+4             ; 1246
                    54              53   D0 0016F          MOVL     VALUE_LOCATION, VALUE_DESC_ADDR              ; 1247
                    15         1A   AE   91 00172          CMPB     VALUE_DESCR+2, #21                           ; 1248
                                    3F   12 00176          BNEQ     28$                                          ;
              20    AE         08   A3   90 00178          MOVB     8(VALUE_LOCATION), VALUE_DESCR+8             ; 1253
                                    38   11 0017D          BRB      28$                                          ; 1235
                    50         04   AC   D0 0017F  24$:    MOVL     VALUE, R0                                    ; 1258
                    0E         02   A0   91 00183          CMPB     2(R0), #14                                   ;
                                    05   13 00187          BEQL     25$                                          ;
                               02   A0   95 00189          TSTB     2(R0)                                        ; 1259
                                    06   12 0018C          BNEQ     26$                                          ;
                    51         10   AC   D0 0018E  25$:    MOVL     STR_LENGTH, R1                               ; 1260
                                    03   11 00192          BRB      27$                                          ;
                    51              60   3C 00194  26$:    MOVZWL   (R0), R1                                     ; 1261
              18    AE              51   B0 00197  27$:    MOVW     R1, VALUE_DESCR                              ; 1258
              1A    AE         02   A0   90 0019B          MOVB     2(R0), VALUE_DESCR+2                         ; 1262
              1B    AE              01   90 001A0          MOVB     #1, VALUE_DESCR+3                            ; 1263
              1C    AE              53   D0 001A4          MOVL     VALUE_LOCATION, VALUE_DESCR+4                ; 1264
```

```
                    54        18   AE   9E 001A8          MOVAB   VALUE_DESCR, VALUE_DESC_ADDR                ; 1265
                    15        1A   AE   91 001AC          CMPB    VALUE_DESCR+2, #21                          ; 1266
                                   05   12 001B0          BNEQ    28$
                 20 AE        08   A6   90 001B2          MOVB    8(R6), VALUE_DESCR+8                        ; 1271
                    52        04   AC   D0 001B7 28$:     MOVL    VALUE, R2                                   ; 1280
                 02 A2        1A   AE   91 001BB          CMPB    VALUE_DESCR+2, 2(R2)
                                   0B   13 001C0          BEQL    29$
                    7E        00G  8F   9A 001C2          MOVZBL  #BAS$K_DATTYPERR, -(SP)
           00000000G 00             01   FB 001C6          CALLS   #1, BAS$$STOP
                 BF 8F        03   A6   91 001CD 29$:     CMPB    3(R6), #191                                 ; 1286
                                   03   13 001D2          BEQL    30$
                                00C7   31 001D4          BRW     38$
                    55             58   D0 001D7 30$:     MOVL    R8, SAVE_LENGTH                             ; 1297
                    50        02   A6   9E 001DA          MOVAB   2(R6), R0                                   ; 1298
                    15             50   D1 001DE          CMPL    R0, #21
                                   0B   12 001E1          BNEQ    31$
                    50        18   AE   3C 001E3          MOVZWL  VALUE_DESCR, R0                             ; 1299
                    50             02   C6 001E7          DIVL2   #2, R0
                                   50   D6 001EA          INCL    R0
                                   04   11 001EC          BRB     32$
                    50        18   AE   3C 001EE 31$:     MOVZWL  VALUE_DESCR, R0                             ; 1300
                    66             50   B0 001F2 32$:     MOVW    R0, (R6)                                    ; 1298
                    18        02   A6   91 001F5          CMPB    2(R6), #24                                  ; 1302
                                   0B   12 001F9          BNEQ    33$
                    7E        00G  8F   9A 001FB          MOVZBL  #BAS$K_NOTIMP, -(SP)
           00000000G 00             01   FB 001FF          CALLS   #1, BAS$$STOP
                    0E        1A   AE   91 00206 33$:     CMPB    VALUE_DESCR+2, #14                          ; 1309
                                   05   13 0020A          BEQL    34$
                             1A    AE   95 0020C          TSTB    VALUE_DESCR+2                               ; 1310
                                   7B   12 0020F          BNEQ    36$
                 10 AE 020E0000    8F   D0 00211 34$:     MOVL    #34471936, TEMP_DESC                        ; 1321
                             14    AE   D4 00219          CLRL    TEMP_DESC+4                                 ; 1324
                                   52   DD 0021C          PUSHL   R2                                          ; 1325
                             14    AE   9F 0021E          PUSHAB  TEMP_DESC
           00000000G 00             02   FB 00221          CALLS   #2, STR$COPY_DX
                    50        18   AE   3C 00228          MOVZWL  VALUE_DESCR, SPACES_COUNT                   ; 1329
                    51        10   AE   3C 0022C          MOVZWL  TEMP_DESC, R1
                    50             51   C2 00230          SUBL2   R1, SPACES_COUNT
                                   3D   15 00233          BLEQ    35$                                         ; 1331
                 0B AE 020E0001    8F   D0 00235          MOVL    #34471937, SPACES_DESC                      ; 1338
                             0C    AE   D4 0023D          CLRL    SPACES_DESC+4                               ; 1341
                 04 AE             20   D0 00240          MOVL    #32, 4(SP)                                  ; 1342
                             04    AE   9F 00244          PUSHAB  4(SP)
                 04 AE             50   D0 00247          MOVL    SPACES_COUNT, 4(SP)
                             04    AE   9F 0024B          PUSHAB  4(SP)
                             10    AE   9F 0024E          PUSHAB  SPACES_DESC
           00000000G 00             03   FB 00251          CALLS   #3, STR$DUPL_CHAR
                             08    AE   9F 00258          PUSHAB  SPACES_DESC                                 ; 1343
                             14    AE   9F 0025B          PUSHAB  TEMP_DESC
                             18    AE   9F 0025E          PUSHAB  TEMP_DESC
           00000000G 00             03   FB 00261          CALLS   #3, STR$CONCAT
                             08    AE   9F 00268          PUSHAB  SPACES_DESC                                 ; 1344
           00000000G 00             01   FB 0026B          CALLS   #1, STR$FREE1_DX
                             14    AE   DD 00272 35$:     PUSHL   TEMP_DESC+4                                 ; 1350
                                   53   DD 00275          PUSHL   VALUE_LOCATION
                                   56   DD 00277          PUSHL   R6
           00000000G 00             03   FB 00279          CALLS   #3, BAS$$VA_STORE
```

```
                                  10  AE  9F 00280        PUSHAB  TEMP_DESC                       ; 1354
                 00000000G  00        01  FB 00283        CALLS   #1, STR$FREE1_DX                ;
                                      0E  11 0028A        BRB     37$                             ; 1309
                                  04  A2  DD 0028C 36$:   PUSHL   4(R2)                           ; 1358
                                      53  DD 0028F        PUSHL   VALUE_LOCATION                  ;
                                      56  DD 00291        PUSHL   R6                              ;
                 00000000G  00        03  FB 00293        CALLS   #3, BAS$$VA_STORE               ;
                            66        55  B0 0029A 37$:   MOVW    SAVE_LENGTH, (R6)               ; 1365
                                      04 0029D            RET                                     ; 1286
                            04    03  A6  91 0029E 38$:   CMPB    3(R6), #4                       ; 1370
                                      0B  13 002A2        BEQL    39$                             ;
                            7E   00G  8F  9A 002A4        MOVZBL  #BAS$K_NOTIMP, -(SP)            ;
                 00000000G  00        01  FB 002A8        CALLS   #1, BAS$$STOP                   ;
                      1C    00    02  A2  8F 002AF 39$:   CASEB   2(R2), #0, #28                  ; 1377
     003A        003A       003A    0094     002B4 40$:   .WORD   49$-40$,-                       ;
     004C        0046       003A    003A     002BC                41$-40$,-                       ;
     0067        0058       003A    0052     002C4                41$-40$,-                       ;
     003A        0094       003A    003A     002CC                41$-40$,-                       ;
     003A        003A       003A    003A     002D4                41$-40$,-                       ;
     003A        003A       00A0    003A     002DC                41$-40$,-                       ;
     0076        003A       003A    003A     002E4                42$-40$,-                       ;
                                    0085     002EC                43$-40$,-                       ;
                                                                  44$-40$,-                       ;
                                                                  41$-40$,-                       ;
                                                                  45$-40$,-                       ;
                                                                  46$-40$,-                       ;
                                                                  41$-40$,-                       ;
                                                                  41$-40$,-                       ;
                                                                  49$-40$,-                       ;
                                                                  41$-40$,-                       ;
                                                                  41$-40$,-                       ;
                                                                  41$-40$,-                       ;
                                                                  41$-40$,-                       ;
                                                                  41$-40$,-                       ;
                                                                  41$-40$,-                       ;
                                                                  50$-40$,-                       ;
                                                                  41$-40$,-                       ;
                                                                  41$-40$,-                       ;
                                                                  41$-40$,-                       ;
                                                                  41$-40$,-                       ;
                                                                  41$-40$,-                       ;
                                                                  47$-40$,-                       ;
                                                                  48$-40$                         ;
                            7E   00G  8F  9A 002EE 41$:   MOVZBL  #BAS$K_DATTYPERR, -(SP)         ; 1419
                 00000000G  00        01  FB 002F2        CALLS   #1, BAS$$STOP                   ;
                                      04 002F9            RET                                     ;
                      1C    BE    04  B2  90 002FA 42$:   MOVB    @4(R2), @VALUE_DESCR+4          ; 1382
                                      04 002FF            RET                                     ;
                      1C    BE    04  B2  B0 00300 43$:   MOVW    @4(R2), @VALUE_DESCR+4          ; 1386
                                      04 00305            RET                                     ;
                      1C    BE    04  B2  D0 00306 44$:   MOVL    @4(R2), @VALUE_DESCR+4          ; 1390
                                      04 0030B            RET                                     ;
                            51    1C  AE  D0 0030C 45$:   MOVL    VALUE_DESCR+4, R1               ; 1393
                            50    04  A2  D0 00310        MOVL    4(R2), R0                       ;
                              00000000G  00  16 00314     JSB     BAS$$COPY_F_R1                  ;
                                      04 0031A            RET                                     ;
                            51    1C  AE  D0 0031B 46$:   MOVL    VALUE_DESCR+4, R1               ; 1396
```

```
                                50        04    A2  DO 0031F          MOVL    4(R2), R0                                       ;
                            00000000G     00  16  00323              JSB     BAS$$COPY_D_R1                                  ;
                                              04  00329              RET                                                     ;
                                51        1C    AE  DO 0032A 47$:    MOVL    VALUE_DESCR+4, R1                               ; 1399
                                50        04    A2  DO 0032E         MOVL    4(R2), R0                                       ;
                            00000000G     00  16  00332              JSB     BAS$$COPY_G_R1                                  ;
                                              04  00338              RET                                                     ;
                                51        1C    AE  DO 00339 48$:    MOVL    VALUE_DESCR+4, R1                               ; 1402
                                50        04    A2  DO 0033D         MOVL    4(R2), R0                                       ;
                            00000000G     00  16  00341              JSB     BAS$$COPY_H_R3                                  ;
                                              04  00347              RET                                                     ;
                                                  52  DD 00348 49$:    PUSHL   R2                                              ; 1405
                                                  54  DD 0034A         PUSHL   VALUE_DESC_ADDR                                 ;
                        00000000G     00          02  FB 0034C         CALLS   #2, STR$COPY_DX                                 ;
                                                  04  00353          RET                                                     ;
                                50        08    A2  98 00354 50$:    CVTBL   8(R2), COUNT                                    ; 1413
                                51        20    AE  98 00358         CVTBL   VALUE_DESCR+8, R1                               ;
                                50              51  C2 0035C         SUBL2   R1, COUNT                                       ;
            00          04    B2              62  50  F8 0035F         ASHP    COUNT, (R2), @4(R2), #0, VALUE_DESCR, -         ; 1415
                                        1C    BE        18    AE          00365                 @VALUE_DESCR+4                  ;
                                                  04  00369          RET                                                     ; 1425

; Routine Size:  874 bytes,    Routine Base:  _BAS$CODE + 0645

; 1260          1426  1
```

```
;  1262          1427  1 GLOBAL ROUTINE BAS$STO_FA_RDX (                       ! Store a string by reference
;  1263          1428  1         VALUE_LEN,                                    ! Length of value
;  1264          1429  1         VALUE_ADDR,                                   ! Address of value
;  1265          1430  1         DESCRIP,                                      ! The descriptor to store it
;  1266          1431  1         INDEX1,                                       ! First index
;  1267          1432  1         INDEX2                                        ! Optional second index
;  1268          1433  1     ) : NOVALUE =
;  1269          1434  1
;  1270          1435  1 !++
;  1271          1436  1 ! FUNCTIONAL DESCRIPTION:
;  1272          1437  1 !
;  1273          1438  1 !       Store a string in an array or virtual array.  The string is
;  1274          1439  1 !       passed by reference.  The compiler uses this entry point
;  1275          1440  1 !       to avoid creating a descriptor for a string constant.
;  1276          1441  1 !
;  1277          1442  1 ! FORMAL PARAMETERS:
;  1278          1443  1 !
;  1279          1444  1 !       VALUE_LEN.rl.v  The length of the string to be stored
;  1280          1445  1 !       VALUE_ADDR.rt.r The address of the string to be stored
;  1281          1446  1 !       DESCRIP.rx.da   The descriptor of the array or virtual array
;  1282          1447  1 !       INDEX1.rl.v     The first index into the array.
;  1283          1448  1 !       INDEX2.rl.v     The second index.  This is optional.
;  1284          1449  1 !
;  1285          1450  1 ! IMPLICIT INPUTS:
;  1286          1451  1 !
;  1287          1452  1 !       NONE
;  1288          1453  1 !
;  1289          1454  1 ! IMPLICIT OUTPUTS:
;  1290          1455  1 !
;  1291          1456  1 !       NONE
;  1292          1457  1 !
;  1293          1458  1 ! ROUTINE VALUE:
;  1294          1459  1 ! COMPLETION CODES:
;  1295          1460  1 !
;  1296          1461  1 !       NONE
;  1297          1462  1 !
;  1298          1463  1 ! SIDE EFFECTS:
;  1299          1464  1 !
;  1300          1465  1 !       Signals if an error is encountered.
;  1301          1466  1 !
;  1302          1467  1 !--
;  1303          1468  1
;  1304          1469  2     BEGIN
;  1305          1470  2
;  1306          1471  2     BUILTIN
;  1307          1472  2         ACTUALCOUNT;
;  1308          1473  2
;  1309          1474  2     LOCAL
;  1310          1475  2         VALUE : BLOCK [8, BYTE];                    ! Build descriptor here
;  1311          1476  2
;  1312          1477  2     VALUE [DSC$W_LENGTH] = .VALUE_LEN;
;  1313          1478  2     VALUE [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  1314          1479  2     VALUE [DSC$B_CLASS] = DSC$K_CLASS_S;
;  1315          1480  2     VALUE [DSC$A_POINTER] = .VALUE_ADDR;
;  1316          1481  2
;  1317          1482  3     IF (ACTUALCOUNT () LSS 5)
;  1318          1485  2     THEN
```

```
;  1319          1484  2          BAS$STORE_BFA (VALUE, .DESCRIP, .INDEX1)
;  1320          1485  2      ELSE
;  1321          1486  2          BAS$STORE_BFA (VALUE,
;  1322          1487  2              .DESCRIP, .INDEX1, .INDEX2);
;  1323          1488  2
;  1324          1489  1      END;                                    ! end of BAS$STO_FA_RDX



                                          0000 00000          .ENTRY  BAS$STO_FA_RDX, Save nothing                    ; 1427
                                5E        08   C2 00002          SUBL2   #8, SP                                          ;
                                6E    04  AC   B0 00005          MOVW    VALUE_LEN, VALUE                                ; 1477
                            02  AE  010E  8F   B0 00009          MOVW    #270, VALUE+2                                   ; 1478
                            04  AE    08  AC   D0 0000F          MOVL    VALUE_ADDR, VALUE+4                             ; 1480
                                05        6C   91 00014          CMPB    (AP), #5                                        ;
                                          0D   1E 00017          BGEQU   1$                                              ;
                                7E    0C  AC   7D 00019          MOVQ    DESCRIP, -(SP)                                  ; 1484
                                    08  AE   9F 0001D          PUSHAB  VALUE                                           ;
                            F93E    CF        03   FB 00020          CALLS   #3, BAS$STORE_BFA                               ;
                                          04   00025          RET                                                     ;
                                7E    10  AC   7D 00026 1$:      MOVQ    INDEX1, -(SP)                                   ; 1487
                                    0C  AC   DD 0002A          PUSHL   DESCRIP                                         ;
                                    0C  AE   9F 0002D          PUSHAB  VALUE                                           ; 1486
                            F92E    CF        04   FB 00030          CALLS   #4, BAS$STORE_BFA                               ;
                                          04   00035          RET                                                     ; 1489

; Routine Size:  54 bytes,    Routine Base:  _BAS$CODE + 09AF

;  1325          1490  1
```

```
;  1327          1491  1 GLOBAL ROUTINE BAS$FET_FA_W_R8 (                      ! Fetch a word
;  1328          1492  1         DESCRIP : REF BLOCK [8, BYTE],                ! The descriptor to fetch from
;  1329          1493  1         INDEX1,                                       ! First index
;  1330          1494  1         INDEX2                                        ! Second index
;  1331          1495  1     ) : VA_JSB =
;  1332          1496  1
;  1333          1497  1 !++
;  1334          1498  1 ! FUNCTIONAL DESCRIPTION:
;  1335          1499  1 !
;  1336          1500  1 !       Fetch a 16-bit word from an array or virtual array.
;  1337          1501  1 !
;  1338          1502  1 ! FORMAL PARAMETERS:
;  1339          1503  1 !
;  1340          1504  1 !       DESCRIP.rw.da   The descriptor of the array or virtual array
;  1341          1505  1 !       INDEX1.rl.v     The first index into the array
;  1342          1506  1 !       INDEX2.rl.v     The second index into the array
;  1343          1507  1 !
;  1344          1508  1 ! IMPLICIT INPUTS:
;  1345          1509  1 !
;  1346          1510  1 !       NONE
;  1347          1511  1 !
;  1348          1512  1 ! IMPLICIT OUTPUTS:
;  1349          1513  1 !
;  1350          1514  1 !       NONE
;  1351          1515  1 !
;  1352          1516  1 ! ROUTINE VALUE:
;  1353          1517  1 ! COMPLETION CODES:
;  1354          1518  1 !
;  1355          1519  1 !       The word from the array or virtual array
;  1356          1520  1 !
;  1357          1521  1 ! SIDE EFFECTS:
;  1358          1522  1 !
;  1359          1523  1 !       Signals if an error is encountered.
;  1360          1524  1 !
;  1361          1525  1 !--
;  1362          1526  1
;  1363          1527  2     BEGIN
;  1364          1528  2
;  1365          1529  2     LOCAL
;  1366          1530  2         BOUNDS : REF VECTOR,
;  1367          1531  2         MULTIPLIERS : REF VECTOR,
;  1368          1532  2         LOW_INDEX,
;  1369          1533  2         HIGH_INDEX,
;  1370          1534  2         INDEX_INCR,
;  1371          1535  2         VALUE_LOCATION,
;  1372          1536  2         INDEX_VALUE,
;  1373          1537  2         INDEX_NUMBER,
;  1374          1538  2         TEMP_DESCRIP: REF BLOCK[,BYTE];
;  1375          1539  2
;  1376          1540  2 !+
;  1377          1541  2 ! Be sure the array has at least one but no more than two dimensions.
;  1378          1542  2 !-
;  1379          1543  2
;  1380          1544  2     IF ((.DESCRIP [DSC$B_DIMCT] LSSU 1) OR (.DESCRIP [DSC$B_DIMCT] GTRU 2)) THEN BAS$$STOP (BAS$K_ONEOR_TWO)
;  1381          1545  2
;  1382          1546  2 !+
;  1383          1547  2 ! Be sure this array or virtual array holds words.
```

```
; 1384     1548  2 !-
; 1385     1549  2
; 1386     1550  2     IF (.DESCRIP [DSC$B_DTYPE] NEQU DSC$K_DTYPE_W)
; 1387     1551  2     THEN
; 1388     1552  2         IF (.DESCRIP [DSC$B_DTYPE] EQLU DSC$K_DTYPE_DSC)
; 1389     1553  2         THEN
; 1390     1554  2             !+
; 1391     1555  2             ! Special handling for dynamically mapped arrays.
; 1392     1556  2             !-
; 1393     1557  2             BEGIN
; 1394     1558  2
; 1395     1559  3             TEMP_DESCRIP = .DESCRIP [DSC$A_POINTER];
; 1396     1560  4             IF (.TEMP_DESCRIP [DSC$B_DTYPE] NEQU DSC$K_DTYPE_W)
; 1397     1561  3             THEN
; 1398     1562  3                 BAS$$STOP (BAS$K_ARGDONMAT);
; 1399     1563  3
; 1400     1564  2             END
; 1401     1565  2         ELSE
; 1402     1566  2             BAS$$STOP (BAS$K_ARGDONMAT);
; 1403     1567  2
; 1404     1568  2
; 1405     1569  2 !+
; 1406     1570  2 ! The coefficients and bounds must be present
; 1407     1571  2 !-
; 1408     1572  2
; 1409     1573  2     IF ( NOT (.DESCRIP [DSC$V_FL_COEFF] AND .DESCRIP [DSC$V_FL_BOUNDS])) THEN BAS$$STOP (BAS$K_ARGDONMAT);
; 1410     1574  2
; 1411     1575  2     MULTIPLIERS = DESCRIP [DSC$L_M1];
; 1412     1576  2     BOUNDS = DESCRIP [DSC$L_M1] + (%UPVAL*.DESCRIP [DSC$B_DIMCT]);
; 1413     1577  2 !+
; 1414     1578  2 ! Compute the lower and upper index numbers based on how the array
; 1415     1579  2 ! is stored.
; 1416     1580  2 !-
; 1417     1581  2
; 1418     1582  2     IF (.DESCRIP [DSC$V_FL_COLUMN])
; 1419     1583  2     THEN
; 1420     1584  3         BEGIN
; 1421     1585  3         LOW_INDEX = .DESCRIP [DSC$B_DIMCT];
; 1422     1586  3         HIGH_INDEX = 1;
; 1423     1587  3         INDEX_INCR = -1;
; 1424     1588  3         END
; 1425     1589  2     ELSE
; 1426     1590  3         BEGIN
; 1427     1591  3         LOW_INDEX = 1;
; 1428     1592  3         HIGH_INDEX = .DESCRIP [DSC$B_DIMCT];
; 1429     1593  3         INDEX_INCR = 1;
; 1430     1594  2         END;
; 1431     1595  2
; 1432     1596  2     INDEX_NUMBER = .LOW_INDEX - .INDEX_INCR;
; 1433     1597  2 !+
; 1434     1598  2 ! Compute the linear index from the indices provided.
; 1435     1599  2 !-
; 1436     1600  2     VALUE_LOCATION = 0;
; 1437     1601  2
; 1438     1602  2     WHILE ((INDEX_NUMBER = .INDEX_NUMBER + .INDEX_INCR) NEQ (.HIGH_INDEX + .INDEX_INCR)) DO
; 1439     1603  3         BEGIN
; 1440     1604  3         INDEX_VALUE = (IF (.INDEX_NUMBER EQL 1) THEN .INDEX1 ELSE .INDEX2);
```

```
;  1441          1605  3          IF ((.INDEX_VALUE LSS .BOUNDS [(.INDEX_NUMBER - 1)*2])  !
;  1442          1606  3              OR (.INDEX_VALUE GTR .BOUNDS [((.INDEX_NUMBER - 1)*2) + 1]))
;  1443          1607  4          THEN
;  1444          1608  4              BAS$$STOP (BAS$K_SUBOUTRAN);
;  1445          1609  4
;  1446          1610  4          VALUE_LOCATION = (.VALUE_LOCATION+.MULTIPLIERS [.INDEX_NUMBER - 1]) + .INDEX_VALUE;
;  1447          1611  4          END;
;  1448          1612  3
;  1449          1613  3      VALUE_LOCATION = (.VALUE_LOCATION+.DESCRIP [DSC$W_LENGTH]) + .DESCRIP [DSC$A_A0];
;  1450          1614  3      IF .DESCRIP [DSC$B_DTYPE] EQLU DSC$K_DTYPE_DSC
;  1451          1615  3      THEN
;  1452          1616  3          !+
;  1453          1617  3          ! Special handling for dynamically mapped arrays.
;  1454          1618  3          !-
;  1455          1619  3          BEGIN
;  1456          1620  3
;  1457          1621  3          TEMP_DESCRIP = .VALUE_LOCATION;
;  1458          1622  3          VALUE_LOCATION = .TEMP_DESCRIP [DSC$A_POINTER];
;  1459          1623  3
;  1460          1624  3          END;
;  1461          1625  3
;  1462          1626  3
;  1463          1627  3  !+
;  1464          1628  3  ! Special handling if this is a virtual array.
;  1465          1629  3  !-
;  1466          1630  3
;  1467          1631  3      IF (.DESCRIP [DSC$B_CLASS] EQLU DSC$K_CLASS_BFA)
;  1468          1632  3      THEN
;  1469          1633  3          BEGIN
;  1470          1634  3
;  1471          1635  3          LOCAL
;  1472          1636  3              VALUE;
;  1473          1637  3
;  1474          1638  3          VALUE = 0;
;  1475          1639  3          BAS$$VA_FETCH (.DESCRIP, .VALUE_LOCATION, VALUE);
;  1476          1640  3          RETURN (.VALUE);
;  1477          1641  3          END;
;  1478          1642  3
;  1479          1643  3      IF (.DESCRIP [DSC$B_CLASS] NEQU DSC$K_CLASS_A) THEN BAS$$STOP (BAS$K_NOTIMP);
;  1480          1644  3
;  1481          1645  3  !+
;  1482          1646  3  ! Return the array element as our value.
;  1483          1647  3  !-
;  1484          1648  2      RETURN (.BLOCK [.VALUE_LOCATION, 0, 0, %BPVAL/2, 1]);
;  1485          1649  1      END;                                    ! end of BAS$FET_FA_W_R8


                                5E            10  C2 00000 BAS$FET_FA_W_R8::
                                                            SUBL2   #16, SP                 ; 1491
                                              51  DD 00003          PUSHL   R1
                                55            50  D0 00005          MOVL    R0, R5
                                53       0B   A5  9A 0000B          MOVZBL  11(DESCRIP), R3 ; 1544
                                              05  13 0000C          BEQL    1$
                                02            53  91 0000E          CMPB    R3, #2          ;
```

```
                                  0B     1B 00011       BLEQU   2$                                       ;
                 7E       00G     8F     9A 00013 1$:   MOVZBL  #BAS$K_ONEOR_TWO, -(SP)                  ;
       00000000G 00               01     FB 00017       CALLS   #1, BAS$$STOP                            ;
                 07       02      A5     91 0001E 2$:   CMPB    2(DESCRIP), #7                           ; 1550
                                  1B     13 00022       BEQL    4$                                       ;
                 18       02      A5     91 00024       CMPB    2(DESCRIP), #24                          ; 1552
                                  0A     12 00028       BNEQ    3$                                       ;
                 57       04      A5     D0 0002A       MOVL    4(DESCRIP), TEMP_DESCRIP                 ; 1559
                 07       02      A7     91 0002E       CMPB    2(TEMP_DESCRIP), #7                      ; 1560
                                  0B     13 00032       BEQL    4$                                       ;
                 7E       00G     8F     9A 00034 3$:   MOVZBL  #BAS$K_ARGDONMAT, -(SP)                  ; 1566
       00000000G 00               01     FB 00038       CALLS   #1, BAS$$STOP                            ;
  05          0A A5               06     E1 0003F 4$:   BBC     #6, 10(DESCRIP), 5$                      ; 1573
                          0A      A5     95 00044       TSTB    10(DESCRIP)                              ;
                                  0B     19 00047       BLSS    6$                                       ;
                 7E       00G     8F     9A 00049 5$:   MOVZBL  #BAS$K_ARGDONMAT, -(SP)                  ;
       00000000G 00               01     FB 0004D       CALLS   #1, BAS$$STOP                            ;
              08 AE       14      A5     9E 00054 6$:   MOVAB   20(R5), MULTIPLIERS                      ; 1575
                 56       14 A543        DE 00059       MOVAL   20(DESCRIP)[R3], BOUNDS                  ; 1576
  0C          0A A5               05     E1 0005E       BBC     #5, 10(DESCRIP), 7$                      ; 1582
                 51               53     D0 00063       MOVL    R3, LOW_INDEX                            ; 1585
                 50               01     D0 00066       MOVL    #1, HIGH_INDEX                           ; 1586
              04 AE               01     CE 00069       MNEGL   #1, INDEX_INCR                           ; 1587
                                  0A     11 0006D       BRB     8$                                       ; 1582
                 51               01     D0 0006F 7$:   MOVL    #1, LOW_INDEX                            ; 1591
                 50               53     D0 00072       MOVL    R3, HIGH_INDEX                           ; 1592
              04 AE               01     D0 00075       MOVL    #1, INDEX_INCR                           ; 1593
  54             51       04      AE     C3 00079 8$:   SUBL3   INDEX_INCR, LOW_INDEX, INDEX_NUMBER      ; 1596
                                  53     D4 0007E       CLRL    VALUE_LOCATION                           ; 1600
              0C AE       04 BE40        9E 00080       MOVAB   @INDEX_INCR[HIGH_INDEX], 12(SP)          ; 1602
                 54       04      AE     C0 00086 9$:   ADDL2   INDEX_INCR, INDEX_NUMBER                 ;
              0C AE               54     D1 0008A       CMPL    INDEX_NUMBER, 12(SP)                     ;
                                  3A     13 0008E       BEQL    14$                                      ;
                 01               54     D1 00090       CMPL    INDEX_NUMBER, #1                         ; 1604
                                  05     12 00093       BNEQ    10$                                      ;
                 58               6E     D0 00095       MOVL    INDEX1, INDEX_VALUE                      ;
                                  03     11 00098       BRB     11$                                      ;
                 58               52     D0 0009A 10$:  MOVL    INDEX2, INDEX_VALUE                      ;
  50             54               01     78 0009D 11$:  ASHL    #1, INDEX_NUMBER, R0                     ; 1606
            F8 A640               58     D1 000A1       CMPL    INDEX_VALUE, -8(BOUNDS)[R0]              ;
                                  07     19 000A6       BLSS    12$                                      ;
            FC A640               58     D1 000A8       CMPL    INDEX_VALUE, -4(BOUNDS)[R0]              ; 1607
                                  0B     15 000AD       BLEQ    13$                                      ;
                 7E       00G     8F     9A 000AF 12$:  MOVZBL  #BAS$K_SUBOUTRAN, -(SP)                  ; 1609
       00000000G 00               01     FB 000B3       CALLS   #1, BAS$$STOP                            ;
  51          08 AE               04     C3 000BA 13$:  SUBL3   #4, MULTIPLIERS, R1                      ; 1611
  50             53             6144     C5 000BF       MULL3   (R1)[INDEX_NUMBER], VALUE_LOCATION, R0   ;
  53             50               58     C1 000C4       ADDL3   INDEX_VALUE, R0, VALUE_LOCATION          ;
                                  BC     11 000C8       BRB     9$                                       ; 1602
                 50               65     3C 000CA 14$:  MOVZWL  (DESCRIP), R0                            ; 1614
                 50               53     C4 000CD       MULL2   VALUE_LOCATION, R0                       ;
  53             50       10      A5     C1 000D0       ADDL3   16(DESCRIP), R0, VALUE_LOCATION          ;
                 18       02      A5     91 000D5       CMPB    2(DESCRIP), #24                          ; 1615
                                  07     12 000D9       BNEQ    15$                                      ;
                 57               53     D0 000DB       MOVL    VALUE_LOCATION, TEMP_DESCRIP             ; 1622
                 53       04      A7     D0 000DE       MOVL    4(TEMP_DESCRIP), VALUE_LOCATION          ; 1623
              BF 8F       03      A5     91 000E2 15$:  CMPB    3(DESCRIP), #191                         ; 1631
```

```
                                 17  12 000E7          BNEQ    16$                                                     ;
                              10 AE  D4 000E9          CLRL    VALUE                                                   ; 1638
                              10 AE  9F 000EC          PUSHAB  VALUE                                                   ; 1639
                                 53  DD 000EF          PUSHL   VALUE_LOCATION                                          ;
                                 55  DD 000F1          PUSHL   DESCRIP                                                 ;
                00000000G     00    03  FB 000F3          CALLS   #3, BAS$$VA_FETCH                                       ;
                              50 10 AE  D0 000FA          MOVL    VALUE, R0                                               ; 1640
                                 14  11 000FE          BRB     18$                                                     ;
                              04 03 A5  91 00100 16$:     CMPB    3(DESCRIP), #4                                          ; 1643
                                 0B  13 00104          BEQL    17$                                                     ;
                              7E 00G 8F  9A 00106          MOVZBL  #BAS$K_NOTIMP, -(SP)                                    ;
                00000000G     00    01  FB 0010A          CALLS   #1, BAS$$STOP                                           ;
                              50    63  32 00111 17$:     CVTWL   (VALUE_LOCATION), R0                                    ; 1648
                              5E    14  C0 00114 18$:     ADDL2   #20, SP                                                 ; 1649
                                    05  00117          RSB                                                             ;

; Routine Size:  280 bytes,    Routine Base:  _BAS$CODE + 09E5

; 1486          1650  1
```

```
;  1488          1651  1 GLOBAL ROUTINE BAS$STO_FA_W_R8 (                      ! Store a word
;  1489          1652  1         VALUE,                                        ! The value to store
;  1490          1653  1         DESCRIP : REF BLOCK [8, BYTE],                ! The descriptor to store into
;  1491          1654  1         INDEX1,                                       ! First index
;  1492          1655  1         INDEX2                                        ! Second index
;  1493          1656  1     ) : VA_JSB NOVALUE =
;  1494          1657  1
;  1495          1658  1 !++
;  1496          1659  1 ! FUNCTIONAL DESCRIPTION:
;  1497          1660  1 !
;  1498          1661  1 !       Store a 16-bit word in an array or virtual array.
;  1499          1662  1 !
;  1500          1663  1 ! FORMAL PARAMETERS:
;  1501          1664  1 !
;  1502          1665  1 !       VALUE.rw.v      The value to store
;  1503          1666  1 !       DESCRIP.rw.da   The descriptor of the array or virtual array
;  1504          1667  1 !       INDEX1.rl.v     The first index into the array
;  1505          1668  1 !       INDEX2.rl.v     The second index into the array
;  1506          1669  1 !
;  1507          1670  1 ! IMPLICIT INPUTS:
;  1508          1671  1 !
;  1509          1672  1 !       NONE
;  1510          1673  1 !
;  1511          1674  1 ! IMPLICIT OUTPUTS:
;  1512          1675  1 !
;  1513          1676  1 !       NONE
;  1514          1677  1 !
;  1515          1678  1 ! ROUTINE VALUE:
;  1516          1679  1 ! COMPLETION CODES:
;  1517          1680  1 !
;  1518          1681  1 !       NONE
;  1519          1682  1 !
;  1520          1683  1 ! SIDE EFFECTS:
;  1521          1684  1 !
;  1522          1685  1 !       Signals if an error is encountered.
;  1523          1686  1 !
;  1524          1687  1 !--
;  1525          1688  1
;  1526          1689  2     BEGIN
;  1527          1690  2
;  1528          1691  2     LOCAL
;  1529          1692  2         BOUNDS : REF VECTOR,
;  1530          1693  2         MULTIPLIERS : REF VECTOR,
;  1531          1694  2         LOW_INDEX,
;  1532          1695  2         HIGH_INDEX,
;  1533          1696  2         INDEX_INCR,
;  1534          1697  2         VALUE_LOCATION,
;  1535          1698  2         INDEX_VALUE,
;  1536          1699  2         INDEX_NUMBER;
;  1537          1700  2
;  1538          1701  2 !+
;  1539          1702  2 ! Be sure the array has at least one but no more than two dimensions.
;  1540          1703  2 !-
;  1541          1704  2
;  1542          1705  2     IF ((.DESCRIP [DSC$B_DIMCT] LSSU 1) OR (.DESCRIP [DSC$B_DIMCT] GTRU 2)) THEN BAS$$STOP (BAS$K_ONEOR_TWO)
;  1543          1706  2
;  1544          1707  2 !+
```

```
;  1545          1708  2 !  Be sure this array or virtual array holds words.
;  1546          1709  2 !-
;  1547          1710  2
;  1548          1711  2     IF (.DESCRIP [DSC$B_DTYPE] NEQU DSC$K_DTYPE_W) THEN BAS$$STOP (BAS$K_ARGDONMAT);
;  1549          1712  2
;  1550          1713  2 !+
;  1551          1714  2 !  The coefficients and bounds must be present
;  1552          1715  2 !-
;  1553          1716  2
;  1554          1717  4     IF ( NOT (.DESCRIP [DSC$V_FL_COEFF] AND     !
;  1555          1718  3         .DESCRIP [DSC$V_FL_BOUNDS]))
;  1556          1719  2     THEN
;  1557          1720  2         BAS$$STOP (BAS$K_ARGDONMAT);
;  1558          1721  2
;  1559          1722  2     MULTIPLIERS = DESCRIP [DSC$L_M1];
;  1560          1723  2     BOUNDS = DESCRIP [DSC$L_M1] + (%UPVAL*.DESCRIP [DSC$B_DIMCT]);
;  1561          1724  2 !+
;  1562          1725  2 !  Compute the lower and upper index numbers based on how the array
;  1563          1726  2 !  is stored.
;  1564          1727  2 !-
;  1565          1728  2
;  1566          1729  3     IF (.DESCRIP [DSC$V_FL_COLUMN])
;  1567          1730  2     THEN
;  1568          1731  3         BEGIN
;  1569          1732  3         LOW_INDEX = .DESCRIP [DSC$B_DIMCT];
;  1570          1733  3         HIGH_INDEX = 1;
;  1571          1734  3         INDEX_INCR = -1;
;  1572          1735  3         END
;  1573          1736  2     ELSE
;  1574          1737  3         BEGIN
;  1575          1738  3         LOW_INDEX = 1;
;  1576          1739  3         HIGH_INDEX = .DESCRIP [DSC$B_DIMCT];
;  1577          1740  3         INDEX_INCR = 1;
;  1578          1741  3         END;
;  1579          1742  2
;  1580          1743  2     INDEX_NUMBER = .LOW_INDEX - .INDEX_INCR;
;  1581          1744  2 !+
;  1582          1745  2 !  Compute the linear index from the indices provided.
;  1583          1746  2 !-
;  1584          1747  2     VALUE_LOCATION = 0;
;  1585          1748  2
;  1586          1749  2     WHILE ((INDEX_NUMBER = .INDEX_NUMBER + .INDEX_INCR) NEQ (.HIGH_INDEX + .INDEX_INCR)) DO
;  1587          1750  3         BEGIN
;  1588          1751  3         INDEX_VALUE = (IF (.INDEX_NUMBER EQL 1) THEN .INDEX1 ELSE .INDEX2);
;  1589          1752  3
;  1590          1753  5         IF ((.INDEX_VALUE LSS .BOUNDS [(.INDEX_NUMBER - 1)*2])  !
;  1591          1754  4             OR (.INDEX_VALUE GTR .BOUNDS [((.INDEX_NUMBER - 1)*2) + 1]))
;  1592          1755  3         THEN
;  1593          1756  3             BAS$$STOP (BAS$K_SUBOUTRAN);
;  1594          1757  3
;  1595          1758  3         VALUE_LOCATION = (.VALUE_LOCATION*.MULTIPLIERS [.INDEX_NUMBER - 1]) + .INDEX_VALUE;
;  1596          1759  3         END;
;  1597          1760  2
;  1598          1761  2     VALUE_LOCATION = (.VALUE_LOCATION*.DESCRIP [DSC$W_LENGTH]) + .DESCRIP [DSC$A_A0];
;  1599          1762  2 !+
;  1600          1763  2 !  Special handling for virtual arrays.
;  1601          1764  2 !-
```

```
;  1602          1765  2
;  1603          1766  3      IF (.DESCRIP [DSC$B_CLASS] EQLU DSC$K_CLASS_BFA)
;  1604          1767  3      THEN
;  1605          1768  3          BEGIN
;  1606          1769  3          BAS$$VA_STORE (.DESCRIP, .VALUE_LOCATION, VALUE);
;  1607          1770  3          END
;  1608          1771  2      ELSE
;  1609          1772  3          BEGIN
;  1610          1773  3
;  1611          1774  3          IF (.DESCRIP [DSC$B_CLASS] NEQU DSC$K_CLASS_A) THEN BAS$$STOP (BAS$K_NOTIMP);
;  1612          1775  3
;  1613          1776  3  !+
;  1614          1777  3  ! Store the value provided into the array
;  1615          1778  3  !-
;  1616          1779  3          BLOCK [.VALUE_LOCATION, 0, 0, %BPVAL/2, 1] = .VALUE;
;  1617          1780  2          END;
;  1618          1781  2
;  1619          1782  1      END;                                        ! end of BAS$STO_FA_W_R8


                                5E              10  C2 00000 BAS$STO_FA_W_R8::
                                                          SUBL2   #16, SP                                 ; 1651
                                56              51  D0 00003          MOVL    R1, R6                     ;
                             0C AE              50  D0 00006          MOVL    R0, VALUE                  ;
                                54         0B   A6  9A 0000A          MOVZBL  11(DESCRIP), R4            ; 1705
                                                05  13 0000E          BEQL    1$                         ;
                                02              54  91 00010          CMPB    R4, #2                     ;
                                                08  1B 00013          BLEQU   2$                         ;
                                7E         00G  8F  9A 00015 1$:      MOVZBL  #BAS$K_ONEOR_TWO, -(SP)    ;
                    00000000G   00              01  FB 00019          CALLS   #1, BAS$$STOP              ;
                                07         02   A6  91 00020 2$:      CMPB    2(DESCRIP), #7             ; 1711
                                                0B  13 00024          BEQL    3$                         ;
                                7E         00G  8F  9A 00026          MOVZBL  #BAS$K_ARGDONMAT, -(SP)    ;
                    00000000G   00              01  FB 0002A          CALLS   #1, BAS$$STOP              ;
                          05    0A   A6                 06  E1 00031 3$:      BBC     #6, 10(DESCRIP), 4$        ; 1717
                                                0A  A6  95 00036          TSTB    10(DESCRIP)                ; 1718
                                                0B  19 00039          BLSS    5$                         ;
                                7E         00G  8F  9A 0003B 4$:      MOVZBL  #BAS$K_ARGDONMAT, -(SP)    ; 1720
                    00000000G   00              01  FB 0003F          CALLS   #1, BAS$$STOP              ;
                             04 AE              14  A6  9E 00046 5$:      MOVAB   20(R6), MULTIPLIERS        ; 1722
                                57              14 A644 DE 0004B          MOVAL   20(DESCRIP)[R4], BOUNDS    ; 1723
                          0B    0A   A6                 05  E1 00050          BBC     #5, 10(DESCRIP), 6$        ; 1729
                                51                      54  D0 00055          MOVL    R4, LOW_INDEX              ; 1732
                                50                      01  D0 00058          MOVL    #1, HIGH_INDEX             ; 1733
                                6E                      01  CE 0005B          MNEGL   #1, INDEX_INCR             ; 1734
                                                        09  11 0005E          BRB     7$                         ; 1729
                                51                      01  D0 00060 6$:      MOVL    #1, LOW_INDEX              ; 1738
                                50                      54  D0 00063          MOVL    R4, HIGH_INDEX             ; 1739
                                6E                      01  D0 00066          MOVL    #1, INDEX_INCR             ; 1740
                          55                    51          6E  C3 00069 7$:      SUBL3   INDEX_INCR, LOW_INDEX, INDEX_NUMBER ; 1743
                                                54          D4 0006D          CLRL    VALUE_LOCATION             ; 1747
                    08    AE                    50          6E  C1 0006F          ADDL3   INDEX_INCR, HIGH_INDEX, 8(SP) ; 1749
                                                55          6E  C0 00074 8$:      ADDL2   INDEX_INCR, INDEX_NUMBER   ;
                                             08 AE          55  D1 00077          CMPL    INDEX_NUMBER, 8(SP)        ;
```

```
                                    3A  13 00078          BEQL    13$                                             ;
                            01      55  D1 0007D          CMPL    INDEX_NUMBER, #1                                ; 1751
                                    05  12 00080          BNEQ    9$                                              ;
                            58      52  D0 00082          MOVL    INDEX1, INDEX_VALUE                             ;
                                    03  11 00085          BRB     10$                                             ;
                            58      53  D0 00087  9$:     MOVL    INDEX2, INDEX_VALUE                             ;
50                          55      01  78 0008A  10$:    ASHL    #1, INDEX_NUMBER, R0                            ; 1753
                    F8 A740         58  D1 0008E          CMPL    INDEX_VALUE, -8(BOUNDS)[R0]                     ;
                                    07  19 00093          BLSS    11$                                             ;
                    FC A740         58  D1 00095          CMPL    INDEX_VALUE, -4(BOUNDS)[R0]                     ; 1754
                                    0B  15 0009A          BLEQ    12$                                             ;
                          7E    00G 8F  9A 0009C  11$:    MOVZBL  #BAS$K_SUBOUTRAN, -(SP)                         ; 1756
            00000000G     00        01  FB 000A0          CALLS   #1, BAS$$STOP                                   ;
51                  04    AE        04  C3 000A7  12$:    SUBL3   #4, MULTIPLIERS, R1                             ; 1758
50                        54    6145  C5 000AC          MULL3   (R1)[INDEX_NUMBER], VALUE_LOCATION, R0          ;
54                        50        58  C1 000B1          ADDL3   INDEX_VALUE, R0, VALUE_LOCATION                 ;
                                    BD  11 000B5          BRB     8$                                              ; 1749
                          50        66  3C 000B7  13$:    MOVZWL  (DESCRIP), R0                                   ; 1761
                          50        54  C4 000BA          MULL2   VALUE_LOCATION, R0                              ;
54                        50    10  A6  C1 000BD          ADDL3   16(DESCRIP), R0, VALUE_LOCATION                 ;
                    BF    8F    03  A6  91 000C2          CMPB    3(DESCRIP), #191                                ; 1766
                                    10  12 000C7          BNEQ    14$                                             ;
                                0C  AE  9F 000C9          PUSHAB  VALUE                                           ; 1769
                                    54  DD 000CC          PUSHL   VALUE_LOCATION                                  ;
                                    56  DD 000CE          PUSHL   DESCRIP                                         ;
            00000000G     00        03  FB 000D0          CALLS   #3, BAS$$VA_STORE                               ;
                                    15  11 000D7          BRB     16$                                             ; 1766
                          04    03  A6  91 000D9  14$:    CMPB    3(DESCRIP), #4                                  ; 1774
                                    0B  13 000DD          BEQL    15$                                             ;
                          7E    00G 8F  9A 000DF          MOVZBL  #BAS$K_NOTIMP, -(SP)                            ;
            00000000G     00        01  FB 000E3          CALLS   #1, BAS$$STOP                                   ;
                          64    0C  AE  B0 000EA  15$:    MOVW    VALUE, (VALUE_LOCATION)                         ; 1779
                          5E        10  C0 000EE  16$:    ADDL2   #16, SP                                         ; 1782
                                    05 000F1              RSB                                                     ;

; Routine Size:  242 bytes,    Routine Base:  _BAS$CODE + 0AFD

; 1620          1783  1
```

```
;  1622          1784  1 GLOBAL ROUTINE BAS$FET_FA_L_R8 (                      ! Fetch a longword
;  1623          1785  1         DESCRIP : REF BLOCK [8, BYTE],               ! The descriptor to fetch from
;  1624          1786  1         INDEX1,                                      ! First index
;  1625          1787  1         INDEX2                                       ! Second index
;  1626          1788  1     ) : VA_JSB =
;  1627          1789  1
;  1628          1790  1 !++
;  1629          1791  1 ! FUNCTIONAL DESCRIPTION:
;  1630          1792  1 !
;  1631          1793  1 !       Fetch a 32-bit longword from an array or virtual array.
;  1632          1794  1 !
;  1633          1795  1 ! FORMAL PARAMETERS:
;  1634          1796  1 !
;  1635          1797  1 !       DESCRIP.rl.da   The descriptor of the array or virtual array
;  1636          1798  1 !       INDEX1.rl.v     The first index into the array
;  1637          1799  1 !       INDEX2.rl.v     The second index into the array
;  1638          1800  1 !
;  1639          1801  1 ! IMPLICIT INPUTS:
;  1640          1802  1 !
;  1641          1803  1 !       NONE
;  1642          1804  1 !
;  1643          1805  1 ! IMPLICIT OUTPUTS:
;  1644          1806  1 !
;  1645          1807  1 !       NONE
;  1646          1808  1 !
;  1647          1809  1 ! ROUTINE VALUE:
;  1648          1810  1 ! COMPLETION CODES:
;  1649          1811  1 !
;  1650          1812  1 !       The longword from the array or virtual array
;  1651          1813  1 !
;  1652          1814  1 ! SIDE EFFECTS:
;  1653          1815  1 !
;  1654          1816  1 !       Signals if an error is encountered.
;  1655          1817  1 !
;  1656          1818  1 !--
;  1657          1819  1
;  1658          1820  2     BEGIN
;  1659          1821  2
;  1660          1822  2     LOCAL
;  1661          1823  2         BOUNDS : REF VECTOR,
;  1662          1824  2         MULTIPLIERS : REF VECTOR,
;  1663          1825  2         LOW_INDEX,
;  1664          1826  2         HIGH_INDEX,
;  1665          1827  2         INDEX_INCR,
;  1666          1828  2         VALUE_LOCATION,
;  1667          1829  2         INDEX_VALUE,
;  1668          1830  2         INDEX_NUMBER,
;  1669          1831  2         TEMP_DESCRIP: REF BLOCK[,BYTE];
;  1670          1832  2
;  1671          1833  2 !+
;  1672          1834  2 ! Be sure the array has at least one but no more than two dimensions.
;  1673          1835  2 !-
;  1674          1836  2
;  1675          1837  2     IF ((.DESCRIP [DSC$B_DIMCT] LSSU 1) OR (.DESCRIP [DSC$B_DIMCT] GTRU 2)) THEN BAS$$STOP (BAS$K_ONEOR_TWO)
;  1676          1838  2
;  1677          1839  2 !+
;  1678          1840  2 ! Be sure this array or virtual array holds longwords.
```

```
;  1679          1841  2 !-
;  1680          1842  2
;  1681          1843  2     IF (.DESCRIP [DSC$B_DTYPE] NEQU DSC$K_DTYPE_L)
;  1682          1844  2     THEN
;  1683          1845  2         IF (.DESCRIP [DSC$B_DTYPE] EQLU DSC$K_DTYPE_DSC)
;  1684          1846  2         THEN
;  1685          1847  2             !+
;  1686          1848  2             ! Special handling for dynamically mapped arrays.
;  1687          1849  2             !-
;  1688          1850  2             BEGIN
;  1689          1851  2
;  1690          1852  2             TEMP_DESCRIP = .DESCRIP [DSC$A_POINTER];
;  1691          1853  2             IF (.TEMP_DESCRIP [DSC$B_DTYPE] NEQU DSC$K_DTYPE_L)
;  1692          1854  2             THEN
;  1693          1855  2                 BAS$$STOP (BAS$K_ARGDONMAT);
;  1694          1856  2
;  1695          1857  2             END
;  1696          1858  2         ELSE
;  1697          1859  2             BAS$$STOP (BAS$K_ARGDONMAT);
;  1698          1860  2
;  1699          1861  2     !+
;  1700          1862  2     ! The coefficients and bounds must be present
;  1701          1863  2     !-
;  1702          1864  2
;  1703          1865  2     IF ( NOT (.DESCRIP [DSC$V_FL_COEFF] AND     !
;  1704          1866  2         .DESCRIP [DSC$V_FL_BOUNDS]))
;  1705          1867  2     THEN
;  1706          1868  2         BAS$$STOP (BAS$K_ARGDONMAT);
;  1707          1869  2
;  1708          1870  2     MULTIPLIERS = DESCRIP [DSC$L_M1];
;  1709          1871  2     BOUNDS = DESCRIP [DSC$L_M1] + (%UPVAL*.DESCRIP [DSC$B_DIMCT]);
;  1710          1872  2     !+
;  1711          1873  2     ! Compute the lower and upper index numbers based on how the array
;  1712          1874  2     ! is stored.
;  1713          1875  2     !-
;  1714          1876  2
;  1715          1877  2     IF (.DESCRIP [DSC$V_FL_COLUMN])
;  1716          1878  2     THEN
;  1717          1879  2         BEGIN
;  1718          1880  2         LOW_INDEX = .DESCRIP [DSC$B_DIMCT];
;  1719          1881  2         HIGH_INDEX = 1;
;  1720          1882  2         INDEX_INCR = -1;
;  1721          1883  2         END
;  1722          1884  2     ELSE
;  1723          1885  2         BEGIN
;  1724          1886  2         LOW_INDEX = 1;
;  1725          1887  2         HIGH_INDEX = .DESCRIP [DSC$B_DIMCT];
;  1726          1888  2         INDEX_INCR = 1;
;  1727          1889  2         END;
;  1728          1890  2
;  1729          1891  2     INDEX_NUMBER = .LOW_INDEX - .INDEX_INCR;
;  1730          1892  2     !+
;  1731          1893  2     ! Compute the linear index from the indices provided.
;  1732          1894  2     !-
;  1733          1895  2     VALUE_LOCATION = 0;
;  1734          1896  2
;  1735          1897  2     WHILE ((INDEX_NUMBER = .INDEX_NUMBER + .INDEX_INCR) NEQ (.HIGH_INDEX + .INDEX_INCR)) DO
```

```
;  1736          1898  1          BEGIN
;  1737          1899  1          INDEX_VALUE = (IF (.INDEX_NUMBER EQL 1) THEN .INDEX1 ELSE .INDEX2);
;  1738          1900  1
;  1739          1901  1          IF ((.INDEX_VALUE LSS .BOUNDS [(.INDEX_NUMBER - 1)*2])  !
;  1740          1902  1              OR (.INDEX_VALUE GTR .BOUNDS [((.INDEX_NUMBER - 1)*2) + 1]))
;  1741          1903  1          THEN
;  1742          1904  1              BAS$$STOP (BAS$K_SUBOUTRAN);
;  1743          1905  1
;  1744          1906  1          VALUE_LOCATION = (.VALUE_LOCATION+.MULTIPLIERS [.INDEX_NUMBER - 1]) + .INDEX_VALUE;
;  1745          1907  1          END;
;  1746          1908  1
;  1747          1909  1      VALUE_LOCATION = (.VALUE_LOCATION+.DESCRIP [DSC$W_LENGTH]) + .DESCRIP [DSC$A_A0];
;  1748          1910  1      IF .DESCRIP [DSC$B_DTYPE] EQLU DSC$K_DTYPE_DSC
;  1749          1911  1      THEN
;  1750          1912  1          !+
;  1751          1913  1          ! Special handling for dynamically mapped arrays.
;  1752          1914  1          !-
;  1753          1915  1          BEGIN
;  1754          1916  1
;  1755          1917  1          TEMP_DESCRIP = .VALUE_LOCATION;
;  1756          1918  1          VALUE_LOCATION = .TEMP_DESCRIP [DSC$A_POINTER];
;  1757          1919  1
;  1758          1920  1          END;
;  1759          1921  1  !+
;  1760          1922  1  ! Special handling for virtual arrays.
;  1761          1923  1  !-
;  1762          1924  1
;  1763          1925  1      IF (.DESCRIP [DSC$B_CLASS] EQLU DSC$K_CLASS_BFA)
;  1764          1926  1      THEN
;  1765          1927  1          BEGIN
;  1766          1928  1
;  1767          1929  1          LOCAL
;  1768          1930  1              VALUE;
;  1769          1931  1
;  1770          1932  1          BAS$$VA_FETCH (.DESCRIP, .VALUE_LOCATION, VALUE);
;  1771          1933  1          RETURN (.VALUE);
;  1772          1934  1          END;
;  1773          1935  1
;  1774          1936  2      IF (.DESCRIP [DSC$B_CLASS] NEQU DSC$K_CLASS_A) THEN BAS$$STOP (BAS$K_NOTIMP);
;  1775          1937  2
;  1776          1938  2  !+
;  1777          1939  2  ! Return the array element as our value.
;  1778          1940  2  !-
;  1779          1941  2      RETURN (.BLOCK [.VALUE_LOCATION, 0, 0, %BPVAL, 1]);
;  1780          1942  1      END;                                    ! end of BAS$FET_FA_L_R8


                                5E            10  C2 00000 BAS$FET_FA_L_R8::
                                                          SUBL2   #16, SP                         ; 1784
                                              51  DD 00003          PUSHL   R1
                                55            50  D0 00005          MOVL    R0, R5
                                53       08   A5  9A 00008          MOVZBL  11(DESCRIP), R3         ; 1837
                                              05  13 0000C          BEQL    1$
                                02            53  91 0000E          CMPB    R3, #2
```

```
                                  0B   1B 00011        BLEQU   2$                                             ;
                  7E       00G    8F   9A 00013 1$:    MOVZBL  #BAS$K_ONEOR_TWO, -(SP)                        ;
         00000000G 00             01   FB 00017        CALLS   #1, BAS$$STOP                                  ;
                  08        02    A5   91 0001E 2$:    CMPB    2(DESCRIP), #8                                 ; 1843
                                  1B   13 00022        BEQL    4$                                             ;
                  18        02    A5   91 00024        CMPB    2(DESCRIP), #24                                ; 1845
                                  0A   12 00028        BNEQ    3$                                             ;
                  57        04    A5   D0 0002A        MOVL    4(DESCRIP), TEMP_DESCRIP                       ; 1852
                  08        02    A7   91 0002E        CMPB    2(TEMP_DESCRIP), #8                            ; 1853
                                  0B   13 00032        BEQL    4$                                             ;
                  7E       00G    8F   9A 00034 3$:    MOVZBL  #BAS$K_ARGDONMAT, -(SP)                        ; 1859
         00000000G 00             01   FB 00038        CALLS   #1, BAS$$STOP                                  ;
05            0A   A5             06   E1 0003F 4$:    BBC     #6, 10(DESCRIP), 5$                            ; 1865
                            0A    A5   95 00044        TSTB    10(DESCRIP)                                    ; 1866
                                  0B   19 00047        BLSS    6$                                             ;
                  7E       00G    8F   9A 00049 5$:    MOVZBL  #BAS$K_ARGDONMAT, -(SP)                        ; 1868
         00000000G 00             01   FB 0004D        CALLS   #1, BAS$$STOP                                  ;
                08 AE       14    A5   9E 00054 6$:    MOVAB   20(R5), MULTIPLIERS                            ; 1870
                   56       14  A543   DE 00059        MOVAL   20(DESCRIP)[R3], BOUNDS                        ; 1871
0C            0A   A5             05   E1 0005E        BBC     #5, 10(DESCRIP), 7$                            ; 1877
                   51             53   D0 00063        MOVL    R3, LOW_INDEX                                  ; 1880
                   50             01   D0 00066        MOVL    #1, HIGH_INDEX                                 ; 1881
              04   AE             01   CE 00069        MNEGL   #1, INDEX_INCR                                 ; 1882
                                  0A   11 0006D        BRB     8$                                             ; 1877
                   51             01   D0 0006F 7$:    MOVL    #1, LOW_INDEX                                  ; 1886
                   50             53   D0 00072        MOVL    R3, HIGH_INDEX                                 ; 1887
              04   AE             01   D0 00075        MOVL    #1, INDEX_INCR                                 ; 1888
54                 51       04    AE   C3 00079 8$:    SUBL3   INDEX_INCR, LOW_INDEX, INDEX_NUMBER            ; 1891
                                  53   D4 0007E        CLRL    VALUE_LOCATION                                 ; 1895
              0C   AE       04  BE40   9E 00080        MOVAB   @INDEX_INCR[HIGH_INDEX], 12(SP)                ; 1897
                   54       04    AE   C0 00086 9$:    ADDL2   INDEX_INCR, INDEX_NUMBER                       ;
              0C   AE             54   D1 0008A        CMPL    INDEX_NUMBER, 12(SP)                           ;
                                  3A   13 0008E        BEQL    14$                                            ;
                   01             54   D1 00090        CMPL    INDEX_NUMBER, #1                               ; 1899
                                  05   12 00093        BNEQ    10$                                            ;
                   58             6E   D0 00095        MOVL    INDEX1, INDEX_VALUE                            ;
                                  03   11 00098        BRB     11$                                            ;
                   58             52   D0 0009A 10$:   MOVL    INDEX2, INDEX_VALUE                            ;
50                 54             01   78 0009D 11$:   ASHL    #1, INDEX_NUMBER, R0                           ; 1901
           F8 A640                58   D1 000A1        CMPL    INDEX_VALUE, -8(BOUNDS)[R0]                    ;
                                  07   19 000A6        BLSS    12$                                            ;
           FC A640                58   D1 000A8        CMPL    INDEX_VALUE, -4(BOUNDS)[R0]                    ; 1902
                                  0B   15 000AD        BLEQ    13$                                            ;
                  7E       00G    8F   9A 000AF 12$:   MOVZBL  #BAS$K_SUBOUTRAN, -(SP)                        ; 1904
         00000000G 00             01   FB 000B3        CALLS   #1, BAS$$STOP                                  ;
51            08   AE             04   C3 000BA 13$:   SUBL3   #4, MULTIPLIERS, R1                            ; 1906
50                 53           6144   C5 000BF        MULL3   (R1)[INDEX_NUMBER], VALUE_LOCATION, R0         ;
53                 50             58   C1 000C4        ADDL3   INDEX_VALUE, R0, VALUE_LOCATION                ;
                                  BC   11 000C8        BRB     9$                                             ; 1897
                   50             65   3C 000CA 14$:   MOVZWL  (DESCRIP), R0                                  ; 1909
                   50             53   C4 000CD        MULL2   VALUE_LOCATION, R0                             ;
53                 50       10    A5   C1 000D0        ADDL3   16(DESCRIP), R0, VALUE_LOCATION                ;
                   18       02    A5   91 000D5        CMPB    2(DESCRIP), #24                                ; 1910
                                  07   12 000D9        BNEQ    15$                                            ;
                   57             53   D0 000DB        MOVL    VALUE_LOCATION, TEMP_DESCRIP                   ; 1917
                   53       04    A7   D0 000DE        MOVL    4(TEMP_DESCRIP), VALUE_LOCATION                ; 1918
              BF   8F       03    A5   91 000E2 15$:   CMPB    3(DESCRIP), #191                               ; 1925
```

```
                                     14  12 000E7          BNEQ    16$                                             ;
                                 10  AE  9F 000E9          PUSHAB  VALUE                                           ; 1932
                                     53  DD 000EC          PUSHL   VALUE_LOCATION                                  ;
                                     55  DD 000EE          PUSHL   DESCRIP                                         ;
                          00000000G  00  03  FB 000F0          CALLS   #3, BAS$$VA_FETCH                           ;
                                     50  10  AE  D0 000F7          MOVL    VALUE, R0                               ; 1933
                                         14  11 000FB          BRB     18$                                     ;
                                     04  03  A5  91 000FD 16$:     CMPB    3(DESCRIP), #4                          ; 1936
                                         0B  13 00101          BEQL    17$                                     ;
                                     7E  00G 8F  9A 00103          MOVZBL  #BAS$K_NOTIMP, -(SP)                    ;
                          00000000G  00      01  FB 00107          CALLS   #1, BAS$$STOP                           ;
                                     50      63  D0 0010E 17$:     MOVL    (VALUE_LOCATION), R0                    ; 1941
                                     5E      14  C0 00111 18$:     ADDL2   #20, SP                                 ; 1942
                                             05 00114          RSB                                             ;

; Routine Size:  277 bytes,    Routine Base:  _BAS$CODE + 0BEF

; 1781          1943  1
```

```
;  1783          1944  1 GLOBAL ROUTINE BAS$STO_FA_L_R8 (                      ! Store a longword
;  1784          1945  1         VALUE,                                       ! The value to store
;  1785          1946  1         DESCRIP : REF BLOCK [8, BYTE],               ! The descriptor to store into
;  1786          1947  1         INDEX1,                                      ! First index
;  1787          1948  1         INDEX2                                       ! Second index
;  1788          1949  1     ) : VA_JSB NOVALUE =
;  1789          1950  1
;  1790          1951  1 !++
;  1791          1952  1 ! FUNCTIONAL DESCRIPTION:
;  1792          1953  1 !
;  1793          1954  1 !       Store a 32-bit longword in an array or virtual array.
;  1794          1955  1 !
;  1795          1956  1 ! FORMAL PARAMETERS:
;  1796          1957  1 !
;  1797          1958  1 !       VALUE.rl.v      The value to store
;  1798          1959  1 !       DESCRIP.rl.da   The descriptor of the array or virtual array
;  1799          1960  1 !       INDEX1.rl.v     The first index into the array
;  1800          1961  1 !       INDEX2.rl.v     The second index into the array
;  1801          1962  1 !
;  1802          1963  1 ! IMPLICIT INPUTS:
;  1803          1964  1 !
;  1804          1965  1 !       NONE
;  1805          1966  1 !
;  1806          1967  1 ! IMPLICIT OUTPUTS:
;  1807          1968  1 !
;  1808          1969  1 !       NONE
;  1809          1970  1 !
;  1810          1971  1 ! ROUTINE VALUE:
;  1811          1972  1 ! COMPLETION CODES:
;  1812          1973  1 !
;  1813          1974  1 !       NONE
;  1814          1975  1 !
;  1815          1976  1 ! SIDE EFFECTS:
;  1816          1977  1 !
;  1817          1978  1 !       Signals if an error is encountered.
;  1818          1979  1 !
;  1819          1980  1 !--
;  1820          1981  1
;  1821          1982  2     BEGIN
;  1822          1983  2
;  1823          1984  2     LOCAL
;  1824          1985  2         BOUNDS : REF VECTOR,
;  1825          1986  2         MULTIPLIERS : REF VECTOR,
;  1826          1987  2         LOW_INDEX,
;  1827          1988  2         HIGH_INDEX,
;  1828          1989  2         INDEX_INCR,
;  1829          1990  2         VALUE_LOCATION,
;  1830          1991  2         INDEX_VALUE,
;  1831          1992  2         INDEX_NUMBER;
;  1832          1993  2
;  1833          1994  2 !+
;  1834          1995  2 ! Be sure the array has at least one but no more than two dimensions.
;  1835          1996  2 !-
;  1836          1997  2
;  1837          1998  2     IF ((.DESCRIP [DSC$B_DIMCT] LSSU 1) OR (.DESCRIP [DSC$B_DIMCT] GTRU 2)) THEN BAS$$STOP (BAS$K_ONEOR_TWO)
;  1838          1999  2
;  1839          2000  2 !+
```

```
;  1840          2001  2 ! Be sure this array or virtual array holds longwords.
;  1841          2002  2 !-
;  1842          2003  2
;  1843          2004  2     IF (.DESCRIP [DSC$B_DTYPE] NEQU DSC$K_DTYPE_L) THEN BAS$$STOP (BAS$K_ARGDONMAT);
;  1844          2005  2
;  1845          2006  2 !+
;  1846          2007  2 ! The coefficients and bounds must be present
;  1847          2008  2 !-
;  1848          2009  2
;  1849          2010  4     IF ( NOT (.DESCRIP [DSC$V_FL_COEFF] AND     !
;  1850          2011  3         .DESCRIP [DSC$V_FL_BOUNDS]))
;  1851          2012  2     THEN
;  1852          2013  2         BAS$$STOP (BAS$K_ARGDONMAT);
;  1853          2014  2
;  1854          2015  2     MULTIPLIERS = DESCRIP [DSC$L_M1];
;  1855          2016  2     BOUNDS = DESCRIP [DSC$L_M1] + (%UPVAL*.DESCRIP [DSC$B_DIMCT]);
;  1856          2017  2 !+
;  1857          2018  2 ! Compute the lower and upper index numbers based on how the array
;  1858          2019  2 ! is stored.
;  1859          2020  2 !-
;  1860          2021  2
;  1861          2022  3     IF (.DESCRIP [DSC$V_FL_COLUMN])
;  1862          2023  2     THEN
;  1863          2024  3         BEGIN
;  1864          2025  3         LOW_INDEX = .DESCRIP [DSC$B_DIMCT];
;  1865          2026  3         HIGH_INDEX = 1;
;  1866          2027  3         INDEX_INCR = -1;
;  1867          2028  3         END
;  1868          2029  2     ELSE
;  1869          2030  3         BEGIN
;  1870          2031  3         LOW_INDEX = 1;
;  1871          2032  3         HIGH_INDEX = .DESCRIP [DSC$B_DIMCT];
;  1872          2033  3         INDEX_INCR = 1;
;  1873          2034  3         END;
;  1874          2035  2
;  1875          2036  2     INDEX_NUMBER = .LOW_INDEX - .INDEX_INCR;
;  1876          2037  2 !+
;  1877          2038  2 ! Compute the linear index from the indices provided.
;  1878          2039  2 !-
;  1879          2040  2     VALUE_LOCATION = 0;
;  1880          2041  2
;  1881          2042  2     WHILE ((INDEX_NUMBER = .INDEX_NUMBER + .INDEX_INCR) NEQ (.HIGH_INDEX + .INDEX_INCR)) DO
;  1882          2043  3         BEGIN
;  1883          2044  3         INDEX_VALUE = (IF (.INDEX_NUMBER EQL 1) THEN .INDEX1 ELSE .INDEX2);
;  1884          2045  3
;  1885          2046  5         IF ((.INDEX_VALUE LSS .BOUNDS [(.INDEX_NUMBER - 1)*2])  !
;  1886          2047  4             OR (.INDEX_VALUE GTR .BOUNDS [((.INDEX_NUMBER - 1)*2) + 1]))
;  1887          2048  3         THEN
;  1888          2049  3             BAS$$STOP (BAS$K_SUBOUTRAN);
;  1889          2050  3
;  1890          2051  3         VALUE_LOCATION = (.VALUE_LOCATION*.MULTIPLIERS [.INDEX_NUMBER - 1]) + .INDEX_VALUE;
;  1891          2052  3         END;
;  1892          2053  2
;  1893          2054  2     VALUE_LOCATION = (.VALUE_LOCATION*.DESCRIP [DSC$W_LENGTH]) + .DESCRIP [DSC$A_A0];
;  1894          2055  2 !+
;  1895          2056  2 ! Special handling for virtual arrays.
;  1896          2057  2 !-
```

```
;  1897          2058  2
;  1898          2059  2      IF (.DESCRIP [DSC$B_CLASS] EQLU DSC$K_CLASS_BFA)
;  1899          2060  2      THEN
;  1900          2061  2          BEGIN
;  1901          2062  2          BAS$$VA_STORE (.DESCRIP, .VALUE_LOCATION, VALUE);
;  1902          2063  2          END
;  1903          2064  2      ELSE
;  1904          2065  2          BEGIN
;  1905          2066  2
;  1906          2067  2          IF (.DESCRIP [DSC$B_CLASS] NEQU DSC$K_CLASS_A) THEN BAS$$STOP (BAS$K_NOTIMP);
;  1907          2068  2
;  1908          2069  2      !+
;  1909          2070  2      ! Store the value provided into the array
;  1910          2071  2      !-
;  1911          2072  2          BLOCK [.VALUF_LOCATION, 0, 0, %BPVAL, 1] = .VALUE;
;  1912          2073  2          END;
;  1913          2074  2
;  1914          2075  1      END;                                    ! end of BAS$STO_FA_L_R8


                                5E            10  C2 00000 BAS$STO_FA_L_R8::
                                                            SUBL2   #16, SP                                 ; 1944
                                56            51  D0 00003          MOVL    R1, R6                      ;
                             0C AE            50  D0 00006          MOVL    R0, VALUE                   ;
                                54       0B   A6  9A 0000A          MOVZBL  11(DESCRIP), R4                 ; 1998
                                                 05  13 0000E          BEQL    1$                          ;
                                02               54  91 00010          CMPB    R4, #2                      ;
                                                 0B  1B 00013          BLEQU   2$                          ;
                                7E          00G  8F  9A 00015 1$:       MOVZBL  #BAS$K_ONEOR_TWO, -(SP)     ;
                          00000000G         00       01  FB 00019          CALLS   #1, BAS$$STOP               ;
                                08       02   A6  91 00020 2$:       CMPB    2(DESCRIP), #8                  ; 2004
                                                 0B  13 00024          BEQL    3$                          ;
                                7E          00G  8F  9A 00026          MOVZBL  #BAS$K_ARGDONMAT, -(SP)     ;
                          00000000G         00       01  FB 0002A          CALLS   #1, BAS$$STOP               ;
                             05    0A   A6          06  E1 00031 3$:       BBC     #6, 10(DESCRIP), 4$         ; 2010
                                                 0A  A6  95 00036          TSTB    10(DESCRIP)                 ; 2011
                                                 0B  19 00039          BLSS    5$                          ;
                                7E          00G  8F  9A 0003B 4$:       MOVZBL  #BAS$K_ARGDONMAT, -(SP)     ; 2013
                          00000000G         00       01  FB 0003F          CALLS   #1, BAS$$STOP               ;
                                04   AE      14  A6  9E 00046 5$:       MOVAB   20(R6), MULTIPLIERS         ; 2015
                                57      14 A644  DE 0004B          MOVAL   20(DESCRIP)[R4], BOUNDS     ; 2016
                             0B    0A   A6          05  E1 00050          BBC     #5, 10(DESCRIP), 6$         ; 2022
                                51               54  D0 00055          MOVL    R4, LOW_INDEX               ; 2025
                                50               01  D0 00058          MOVL    #1, HIGH_INDEX              ; 2026
                                6E               01  CE 0005B          MNEGL   #1, INDEX_INCR              ; 2027
                                                 09  11 0005E          BRB     7$                          ; 2022
                                51               01  D0 00060 6$:       MOVL    #1, LOW_INDEX               ; 2031
                                50               54  D0 00063          MOVL    R4, HIGH_INDEX              ; 2032
                                6E               01  D0 00066          MOVL    #1, INDEX_INCR              ; 2033
                             55                 51  6E  C3 00069 7$:       SUBL3   INDEX_INCR, LOW_INDEX, INDEX_NUMBER ; 2036
                                                 54  D4 0006D          CLRL    VALUE_LOCATION              ; 2040
                          08 AE               50  6E  C1 0006F          ADDL3   INDEX_INCR, HIGH_INDEX, 8(SP) ; 2042
                                               55  6E  C0 00074 8$:       ADDL2   INDEX_INCR, INDEX_NUMBER    ;
                                08 AE          55  D1 00077          CMPL    INDEX_NUMBER, 8(SP)         ;
```

```
                                    3A  13 0007B          BEQL    13$                                             ;
                          01        55  D1 0007D          CMPL    INDEX_NUMBER, #1                                ; 2044
                                    05  12 00080          BNEQ    9$
                          58        52  D0 00082          MOVL    INDEX1, INDEX_VALUE
                                    03  11 00085          BRB     10$
                          58        53  D0 00087  9$:     MOVL    INDEX2, INDEX_VALUE
                    50    55        01  78 0008A  10$:    ASHL    #1, INDEX_NUMBER, R0                            ; 2046
                FB A740             58  D1 0008E          CMPL    INDEX_VALUE, -8(BOUNDS)[R0]
                                    07  19 00093          BLSS    11$
                FC A740             58  D1 00095          CMPL    INDEX_VALUE, -4(BOUNDS)[R0]                     ; 2047
                                    0B  15 0009A          BLEQ    12$
                          7E  00G   8F  9A 0009C  11$:     MOVZBL  #BAS$K_SUBOUTRAN, -(SP)                         ; 2049
    00000000G             00        01  FB 000A0          CALLS   #1, BAS$$STOP
              51    04    AE        04  C3 000A7  12$:     SUBL3   #4, MULTIPLIERS, R1                             ; 2051
              50          54    6145  C5 000AC          MULL3   (R1)[INDEX_NUMBER], VALUE_LOCATION, R0
              54          50        58  C1 000B1          ADDL3   INDEX_VALUE, R0, VALUE_LOCATION
                                    BD  11 000B5          BRB     8$                                              ; 2042
                          50        66  3C 000B7  13$:     MOVZWL  (DESCRIP), R0                                   ; 2054
                          50        54  C4 000BA          MULL2   VALUE_LOCATION, R0
              54          50  10    A6  C1 000BD          ADDL3   16(DESCRIP), R0, VALUE_LOCATION
                    BF    8F  03    A6  91 000C2          CMPB    3(DESCRIP), #191                                ; 2059
                                    10  12 000C7          BNEQ    14$
                                0C  AE  9F 000C9          PUSHAB  VALUE                                           ; 2062
                                    54  DD 000CC          PUSHL   VALUE_LOCATION
                                    56  DD 000CE          PUSHL   DESCRIP
    00000000G             00        03  FB 000D0          CALLS   #3, BAS$$VA_STORE
                                    15  11 000D7          BRB     16$                                             ; 2059
                          04  03    A6  91 000D9  14$:     CMPB    3(DESCRIP), #4                                  ; 2067
                                    0B  13 000DD          BEQL    15$
                          7E  00G   8F  9A 000DF          MOVZBL  #BAS$K_NOTIMP, -(SP)
    00000000G             00        01  FB 000E3          CALLS   #1, BAS$$STOP
                          64    0C  AE  D0 000EA  15$:     MOVL    VALUE, (VALUE_LOCATION)                         ; 2072
                          5E        10  C0 000EE  16$:     ADDL2   #16, SP                                         ; 2075
                                        05 000F1          RSB                                                     ;

; Routine Size:  242 bytes,    Routine Base:  _BAS$CODE + 0D04

; 1915          2076  1
```

```
;  1917          2077  1 GLOBAL ROUTINE BAS$FET_FA_F_RB (                      ! Fetch a floating number
;  1918          2078  1         DESCRIP : REF BLOCK [8, BYTE],               ! The descriptor to fetch from
;  1919          2079  1         INDEX1,                                      ! First index
;  1920          2080  1         INDEX2                                       ! Second index
;  1921          2081  1     ) : VA_JSB =
;  1922          2082  1
;  1923          2083  1 !++
;  1924          2084  1 ! FUNCTIONAL DESCRIPTION:
;  1925          2085  1 !
;  1926          2086  1 !       Fetch a single precision floating number from an array or virtual array.
;  1927          2087  1 !
;  1928          2088  1 ! FORMAL PARAMETERS:
;  1929          2089  1 !
;  1930          2090  1 !       DESCRIP.rf.da   The descriptor of the array or virtual array
;  1931          2091  1 !       INDEX1.rl.v     The first index into the array
;  1932          2092  1 !       INDEX2.rl.v     The second index into the array
;  1933          2093  1 !
;  1934          2094  1 ! IMPLICIT INPUTS:
;  1935          2095  1 !
;  1936          2096  1 !       NONE
;  1937          2097  1 !
;  1938          2098  1 ! IMPLICIT OUTPUTS:
;  1939          2099  1 !
;  1940          2100  1 !       NONE
;  1941          2101  1 !
;  1942          2102  1 ! ROUTINE VALUE:
;  1943          2103  1 ! COMPLETION CODES:
;  1944          2104  1 !
;  1945          2105  1 !       The number from the array or virtual array
;  1946          2106  1 !
;  1947          2107  1 ! SIDE EFFECTS:
;  1948          2108  1 !
;  1949          2109  1 !       Signals if an error is encountered.
;  1950          2110  1 !
;  1951          2111  1 !--
;  1952          2112  1
;  1953          2113  2     BEGIN
;  1954          2114  2
;  1955          2115  2     LOCAL
;  1956          2116  2         BOUNDS : REF VECTOR,
;  1957          2117  2         MULTIPLIERS : REF VECTOR,
;  1958          2118  2         LOW_INDEX,
;  1959          2119  2         HIGH_INDEX,
;  1960          2120  2         INDEX_INCR,
;  1961          2121  2         VALUE_LOCATION,
;  1962          2122  2         INDEX_VALUE,
;  1963          2123  2         INDEX_NUMBER,
;  1964          2124  2         VALUE,
;  1965          2125  2         TEMP_DESCRIP: REF BLOCK[,BYTE];
;  1966          2126  2
;  1967          2127  2 !+
;  1968          2128  2 ! Be sure the array has at least one but no more than two dimensions.
;  1969          2129  2 !-
;  1970          2130  2
;  1971          2131  2     IF ((.DESCRIP [DSC$B_DIMCT] LSSU 1) OR (.DESCRIP [DSC$B_DIMCT] GTRU 2)) THEN BAS$$STOP (BAS$K_ONEOR_TWO)
;  1972          2132  2
;  1973          2133  2 !+
```

```
;  1974          2134  2 !Be sure this array or virtual array holds floating values.
;  1975          2135  2 !-
;  1976          2136  2
;  1977          2137  2     IF (.DESCRIP [DSC$B_DTYPE] NEQU DSC$K_DTYPE_F)
;  1978          2138  2     THEN
;  1979          2139  2         IF (.DESCRIP [DSC$B_DTYPE] EQLU DSC$K_DTYPE_DSC)
;  1980          2140  2         THEN
;  1981          2141  2             !+
;  1982          2142  2             ! Special handling for dynamically mapped arrays.
;  1983          2143  2             !-
;  1984          2144  2             BEGIN
;  1985          2145  2
;  1986          2146  2             TEMP_DESCRIP = .DESCRIP [DSC$A_POINTER];
;  1987          2147  2             IF (.TEMP_DESCRIP [DSC$B_DTYPE] NEQU DSC$K_DTYPE_F)
;  1988          2148  2             THEN
;  1989          2149  2                 BAS$$STOP (BAS$K_ARGDONMAT);
;  1990          2150  2
;  1991          2151  2             END
;  1992          2152  2         ELSE
;  1993          2153  2             BAS$$STOP (BAS$K_ARGDONMAT);
;  1994          2154  2
;  1995          2155  2 !+
;  1996          2156  2 ! The coefficients and bounds must be present
;  1997          2157  2 !-
;  1998          2158  2
;  1999          2159  2     IF ( NOT (.DESCRIP [DSC$V_FL_COEFF] AND      !
;  2000          2160  2         .DESCRIP [DSC$V_FL_BOUNDS]))
;  2001          2161  2     THEN
;  2002          2162  2         BAS$$STOP (BAS$K_ARGDONMAT);
;  2003          2163  2
;  2004          2164  2     MULTIPLIERS = DESCRIP [DSC$L_M1];
;  2005          2165  2     BOUNDS = DESCRIP [DSC$L_M1] + (%UPVAL*.DESCRIP [DSC$B_DIMCT]);
;  2006          2166  2 !+
;  2007          2167  2 ! Compute the lower and upper index numbers based on how the array
;  2008          2168  2 ! is stored.
;  2009          2169  2 !-
;  2010          2170  2
;  2011          2171  2     IF (.DESCRIP [DSC$V_FL_COLUMN])
;  2012          2172  2     THEN
;  2013          2173  2         BEGIN
;  2014          2174  2         LOW_INDEX = .DESCRIP [DSC$B_DIMCT];
;  2015          2175  2         HIGH_INDEX = 1;
;  2016          2176  2         INDEX_INCR = -1;
;  2017          2177  2         END
;  2018          2178  2     ELSE
;  2019          2179  2         BEGIN
;  2020          2180  2         LOW_INDEX = 1;
;  2021          2181  2         HIGH_INDEX = .DESCRIP [DSC$B_DIMCT];
;  2022          2182  2         INDEX_INCR = 1;
;  2023          2183  2         END;
;  2024          2184  2
;  2025          2185  2     INDEX_NUMBER = .LOW_INDEX - .INDEX_INCR;
;  2026          2186  2 !+
;  2027          2187  2 ! Compute the linear index from the indices provided.
;  2028          2188  2 !-
;  2029          2189  2     VALUE_LOCATION = 0;
;  2030          2190  2
```

```
; 2031          2191  2      WHILE ((INDEX_NUMBER = .INDEX_NUMBER + .INDEX_INCR) NEQ (.HIGH_INDEX + .INDEX_INCR)) DO
; 2032          2192  2          BEGIN
; 2033          2193  2          INDEX_VALUE = (IF (.INDEX_NUMBER EQL 1) THEN .INDEX1 ELSE .INDEX2);
; 2034          2194  2
; 2035          2195  2          IF ((.INDEX_VALUE LSS .BOUNDS [(.INDEX_NUMBER - 1)*2])  !
; 2036          2196  2              OR (.INDEX_VALUE GTR .BOUNDS [((.INDEX_NUMBER - 1)*2) + 1]))
; 2037          2197  2          THEN
; 2038          2198  2              BAS$$STOP (BAS$K_SUBOUTRAN);
; 2039          2199  2
; 2040          2200  2          VALUE_LOCATION = (.VALUE_LOCATION+.MULTIPLIERS [.INDEX_NUMBER - 1]) + .INDEX_VALUE;
; 2041          2201  2          END;
; 2042          2202  2
; 2043          2203  2      VALUE_LOCATION = (.VALUE_LOCATION+.DESCRIP [DSC$W_LENGTH]) + .DESCRIP [DSC$A_A0];
; 2044          2204  2      IF .DESCRIP [DSC$B_DTYPE] EQLU DSC$K_DTYPE_DSC
; 2045          2205  2      THEN
; 2046          2206  2          !+
; 2047          2207  2          ! Special handling for dynamically mapped arrays.
; 2048          2208  2          !-
; 2049          2209  2          BEGIN
; 2050          2210  2
; 2051          2211  2          TEMP_DESCRIP = .VALUE_LOCATION;
; 2052          2212  2          VALUE_LOCATION = .TEMP_DESCRIP [DSC$A_POINTER];
; 2053          2213  2
; 2054          2214  2          END;
; 2055          2215  2  !+
; 2056          2216  2  ! Special handling for virtual arrays.
; 2057          2217  2  !-
; 2058          2218  2
; 2059          2219  2      IF (.DESCRIP [DSC$B_CLASS] EQLU DSC$K_CLASS_BFA)
; 2060          2220  2      THEN
; 2061          2221  2          BEGIN
; 2062          2222  2
; 2063          2223  2          LOCAL
; 2064          2224  2              VALUE;
; 2065          2225  2
; 2066          2226  2          BAS$$VA_FETCH (.DESCRIP, .VALUE_LOCATION, VALUE);
; 2067          2227  2          RETURN (.VALUE);
; 2068          2228  2          END;
; 2069          2229  2
; 2070          2230  2      IF (.DESCRIP [DSC$B_CLASS] NEQU DSC$K_CLASS_A) THEN BAS$$STOP (BAS$K_NOTIMP);
; 2071          2231  2
; 2072          2232  2  !+
; 2073          2233  2  ! Fetch the value using the MOVF instruction to be sure that it is not
; 2074          2234  2  ! a floating reserved operand.
; 2075          2235  2  !-
; 2076          2236  2      BAS$$COPY_F_R1 (.VALUE_LOCATION, VALUE);
; 2077          2237  2  !+
; 2078          2238  2  ! Return the array element as our value.
; 2079          2239  2  !-
; 2080          2240  2      RETURN (.VALUE);
; 2081          2241  1      END;                                        ! end of BAS$FET_FA_F_R8


                               5E            14  C2 00000 BAS$FET_FA_F_R8::
```

```
                                                   SUBL2   #20, SP                                     ; 2077
                          51 DD 00003              PUSHL   R1                                          ;
           54             50 D0 00005              MOVL    R0, R4                                      ;
           53      0B     A4 9A 00008              MOVZBL  11(DESCRIP), R3                             ; 2131
                          05 13 0000C              BEQL    1$                                          ;
           02             53 91 0000E              CMPB    R3, #2                                      ;
                          0B 1B 00011              BLEQU   2$                                          ;
           7E      00G    8F 9A 00013 1$:          MOVZBL  #BAS$K_ONEOR_TWO, -(SP)                     ;
  00000000G 00            01 FB 00017              CALLS   #1, BAS$$STOP                               ;
           0A      02     A4 91 0001E 2$:          CMPB    2(DESCRIP), #10                             ; 2137
                          18 13 00022              BEQL    4$                                          ;
           18      02     A4 91 00024              CMPB    2(DESCRIP), #24                             ; 2139
                          0A 12 00028              BNEQ    3$                                          ;
           56      04     A4 D0 0002A              MOVL    4(DESCRIP), TEMP_DESCRIP                    ; 2146
           0A      02     A6 91 0002E              CMPB    2(TEMP_DESCRIP), #10                        ; 2147
                          0B 13 00032              BEQL    4$                                          ;
           7E      00G    8F 9A 00034 3$:          MOVZBL  #BAS$K_ARGDONMAT, -(SP)                     ; 2153
  00000000G 00            01 FB 00038              CALLS   #1, BAS$$STOP                               ;
05        0A A4           06 E1 0003F 4$:          BBC     #6, 10(DESCRIP), 5$                         ; 2159
                   0A     A4 95 00044              TSTB    10(DESCRIP)                                 ; 2160
                          0B 19 00047              BLSS    6$                                          ;
           7E      00G    8F 9A 00049 5$:          MOVZBL  #BAS$K_ARGDONMAT, -(SP)                     ; 2162
  00000000G 00            01 FB 0004D              CALLS   #1, BAS$$STOP                               ;
        08 AE      14     A4 9E 00054 6$:          MOVAB   20(R4), MULTIPLIERS                         ; 2164
           55      14 A443   DE 00059              MOVAL   20(DESCRIP)[R3], BOUNDS                     ; 2165
0C        0A A4           05 E1 0005E              BBC     #5, 10(DESCRIP), 7$                         ; 2171
           51             53 D0 00063              MOVL    R3, LOW_INDEX                               ; 2174
           50             01 D0 00066              MOVL    #1, HIGH_INDEX                              ; 2175
        04 AE             01 CE 00069              MNEGL   #1, INDEX_INCR                              ; 2176
                          0A 11 0006D              BRB     8$                                          ; 2171
           51             01 D0 0006F 7$:          MOVL    #1, LOW_INDEX                               ; 2180
           50             53 D0 00072              MOVL    R3, HIGH_INDEX                              ; 2181
        04 AE             01 D0 00075              MOVL    #1, INDEX_INCR                              ; 2182
53         51      04     AE C3 00079 8$:          SUBL3   INDEX_INCR, LOW_INDEX, INDEX_NUMBER         ; 2185
                          57 D4 0007E              CLRL    VALUE_LOCATION                              ; 2189
        0C AE      04 BE40   9E 00080              MOVAB   @INDEX_INCR[HIGH_INDEX], 12(SP)             ; 2191
           53      04     AE C0 00086 9$:          ADDL2   INDEX_INCR, INDEX_NUMBER                    ;
        0C AE             53 D1 0008A              CMPL    INDEX_NUMBER, 12(SP)                        ;
                          3A 13 0008E              BEQL    14$                                         ;
           01             53 D1 00090              CMPL    INDEX_NUMBER, #1                            ; 2193
                          05 12 00093              BNEQ    10$                                         ;
           58             6E D0 00095              MOVL    INDEX1, INDEX_VALUE                         ;
                          03 11 00098              BRB     11$                                         ;
           58             52 D0 0009A 10$:         MOVL    INDEX2, INDEX_VALUE                         ;
50         53             01 78 0009D 11$:         ASHL    #1, INDEX_NUMBER, R0                        ; 2195
      FB A540             58 D1 000A1              CMPL    INDEX_VALUE, -8(BOUNDS)[R0]                 ;
                          07 19 000A6              BLSS    12$                                         ;
      FC A540             58 D1 000A8              CMPL    INDEX_VALUE, -4(BOUNDS)[R0]                 ; 2196
                          0B 15 000AD              BLEQ    13$                                         ;
           7E      00G    8F 9A 000AF 12$:         MOVZBL  #BAS$K_SUBOUTRAN, -(SP)                     ; 2198
  00000000G 00            01 FB 000B3              CALLS   #1, BAS$$STOP                               ;
51      08 AE             04 C3 000BA 13$:         SUBL3   #4, MULTIPLIERS, R1                         ; 2200
50         57           6143 C5 000BF              MULL3   (R1)[INDEX_NUMBER], VALUE_LOCATION, R0      ;
57         50             58 C1 000C4              ADDL3   INDEX_VALUE, R0, VALUE_LOCATION             ;
                          BC 11 000C8              BRB     9$                                          ; 2191
           50             64 3C 000CA 14$:         MOVZWL  (DESCRIP), R0                               ; 2203
           50             57 C4 000CD              MULL2   VALUE_LOCATION, R0                          ;
```

```
                    57      50      10  A4  C1 000D0          ADDL3   16(DESCRIP), R0, VALUE_LOCATION
                            18      02  A4  91 000D5          CMPB    2(DESCRIP), #24                         ;2204
                                        07  12 000D9          BNEQ    15$
                            56          57  D0 000DB          MOVL    VALUE_LOCATION, TEMP_DESCRIP            ;2211
                            57      04  A6  D0 000DE          MOVL    4(TEMP_DESCRIP), VALUE_LOCATION         ;2212
                        BF  8F      03  A4  91 000E2 15$:      CMPB    3(DESCRIP), #191                        ;2219
                                        14  12 000E7          BNEQ    16$
                                    10  AE  9F 000E9          PUSHAB  VALUE                                   ;2226
                                  0090  8F  BB 000EC          PUSHR   #^M<R4,R7>
                00000000G           00      03  FB 000F0      CALLS   #3, BAS$$VA_FETCH
                            50      10  AE  D0 000F7          MOVL    VALUE, R0                               ;2227
                                        22  11 000FB          BRB     18$
                            04      03  A4  91 000FD 16$:      CMPB    3(DESCRIP), #4                          ;2230
                                        08  13 00101          BEQL    17$
                            7E      00G 8F  9A 00103          MOVZBL  #BAS$K_NOTIMP, -(SP)
                00000000G           00      01  FB 00107      CALLS   #1, BAS$$STOP
                            51      14  AE  9E 0010E 17$:      MOVAB   VALUE, R1                               ;2236
                            50          57  D0 00112          MOVL    VALUE_LOCATION, R0
                                00000000G   00  16 00115      JSB     BAS$$COPY_F_R1
                            50      14  AE  D0 0011B          MOVL    VALUE, R0                               ;2240
                            5E          18  C0 0011F 18$:      ADDL2   #24, SP                                 ;2241
                                            05 00122          RSB

; Routine Size:  291 bytes,    Routine Base:  _BAS$CODE + 0DF6

; 2082          2242  1
```

```
;  2084          2243  1 GLOBAL ROUTINE BAS$STO_FA_F_R8 (                      ! Store a floating value
;  2085          2244  1         VALUE,                                       ! The value to store
;  2086          2245  1         DESCRIP : REF BLOCK [8, BYTE],               ! The descriptor to store into
;  2087          2246  1         INDEX1,                                      ! First index
;  2088          2247  1         INDEX2                                       ! Second index
;  2089          2248  1     ) : VA_JSB NOVALUE =
;  2090          2249  1
;  2091          2250  1 !++
;  2092          2251  1 ! FUNCTIONAL DESCRIPTION:
;  2093          2252  1 !
;  2094          2253  1 !       Store a single-precision value in an array or virtual array.
;  2095          2254  1 !
;  2096          2255  1 ! FORMAL PARAMETERS:
;  2097          2256  1 !
;  2098          2257  1 !       VALUE.rf.v      The value to store
;  2099          2258  1 !       DESCRIP.rf.da   The descriptor of the array or virtual array
;  2100          2259  1 !       INDEX1.rl.v     The first index into the array
;  2101          2260  1 !       INDEX2.rl.v     The second index into the array
;  2102          2261  1 !
;  2103          2262  1 ! IMPLICIT INPUTS:
;  2104          2263  1 !
;  2105          2264  1 !       NONE
;  2106          2265  1 !
;  2107          2266  1 ! IMPLICIT OUTPUTS:
;  2108          2267  1 !
;  2109          2268  1 !       NONE
;  2110          2269  1 !
;  2111          2270  1 ! ROUTINE VALUE:
;  2112          2271  1 ! COMPLETION CODES:
;  2113          2272  1 !
;  2114          2273  1 !       NONE
;  2115          2274  1 !
;  2116          2275  1 ! SIDE EFFECTS:
;  2117          2276  1 !
;  2118          2277  1 !       Signals if an error is encountered.
;  2119          2278  1 !
;  2120          2279  1 !--
;  2121          2280  1
;  2122          2281  2     BEGIN
;  2123          2282  2
;  2124          2283  2     LOCAL
;  2125          2284  2         BOUNDS : REF VECTOR,
;  2126          2285  2         MULTIPLIERS : REF VECTOR,
;  2127          2286  2         LOW_INDEX,
;  2128          2287  2         HIGH_INDEX,
;  2129          2288  2         INDEX_INCR,
;  2130          2289  2         VALUE_LOCATION,
;  2131          2290  2         INDEX_VALUE,
;  2132          2291  2         INDEX_NUMBER;
;  2133          2292  2
;  2134          2293  2 !+
;  2135          2294  2 ! Be sure the array has at least one but no more than two dimensions.
;  2136          2295  2 !-
;  2137          2296  2
;  2138          2297  2     IF ((.DESCRIP [DSC$B_DIMCT] LSSU 1) OR (.DESCRIP [DSC$B_DIMCT] GTRU 2)) THEN BAS$$STOP (BAS$K_ONEOR_TWO)
;  2139          2298  2
;  2140          2299  2 !+
```

```
;  2141          2300  2 ! Be sure this array or virtual array holds floating values.
;  2142          2301  2 !-
;  2143          2302  2
;  2144          2303  2     IF (.DESCRIP [DSC$B_DTYPE] NEQU DSC$K_DTYPE_F) THEN BAS$$STOP (BAS$K_ARGDONMAT);
;  2145          2304  2
;  2146          2305  2 !+
;  2147          2306  2 ! The coefficients and bounds must be present
;  2148          2307  2 !-
;  2149          2308  2
;  2150          2309  2     IF ( NOT (.DESCRIP [DSC$V_FL_COEFF] AND      !
;  2151          2310  3         .DESCRIP [DSC$V_FL_BOUNDS]))
;  2152          2311  2     THEN
;  2153          2312  3         BAS$$STOP (BAS$K_ARGDONMAT);
;  2154          2313  3
;  2155          2314  2     MULTIPLIERS = DESCRIP [DSC$L_M1];
;  2156          2315  2     BOUNDS = DESCRIP [DSC$L_M1] + (%UPVAL*.DESCRIP [DSC$B_DIMCT]);
;  2157          2316  2 !+
;  2158          2317  2 ! Compute the lower and upper index numbers based on how the array
;  2159          2318  2 ! is stored.
;  2160          2319  2 !-
;  2161          2320  2
;  2162          2321  2     IF (.DESCRIP [DSC$V_FL_COLUMN])
;  2163          2322  2     THEN
;  2164          2323  3         BEGIN
;  2165          2324  3         LOW_INDEX = .DESCRIP [DSC$B_DIMCT];
;  2166          2325  3         HIGH_INDEX = 1;
;  2167          2326  3         INDEX_INCR = -1;
;  2168          2327  3         END
;  2169          2328  2     ELSE
;  2170          2329  3         BEGIN
;  2171          2330  3         LOW_INDEX = 1;
;  2172          2331  3         HIGH_INDEX = .DESCRIP [DSC$B_DIMCT];
;  2173          2332  3         INDEX_INCR = 1;
;  2174          2333  3         END;
;  2175          2334  2
;  2176          2335  2     INDEX_NUMBER = .LOW_INDEX - .INDEX_INCR;
;  2177          2336  2 !+
;  2178          2337  2 ! Compute the linear index from the indices provided.
;  2179          2338  2 !-
;  2180          2339  2     VALUE_LOCATION = 0;
;  2181          2340  2
;  2182          2341  2     WHILE ((INDEX_NUMBER = .INDEX_NUMBER + .INDEX_INCR) NEQ (.HIGH_INDEX + .INDEX_INCR)) DO
;  2183          2342  3         BEGIN
;  2184          2343  3         INDEX_VALUE = (IF (.INDEX_NUMBER EQL 1) THEN .INDEX1 ELSE .INDEX2);
;  2185          2344  3
;  2186          2345  3         IF ((.INDEX_VALUE LSS .BOUNDS [(.INDEX_NUMBER - 1)*2])  !
;  2187          2346  4             OR (.INDEX_VALUE GTR .BOUNDS [((.INDEX_NUMBER - 1)*2) + 1]))
;  2188          2347  3         THEN
;  2189          2348  4             BAS$$STOP (BAS$K_SUBOUTRAN);
;  2190          2349  4
;  2191          2350  3         VALUE_LOCATION = (.VALUE_LOCATION*.MULTIPLIERS [.INDEX_NUMBER - 1]) + .INDEX_VALUE;
;  2192          2351  3         END;
;  2193          2352  2
;  2194          2353  2     VALUE_LOCATION = (.VALUE_LOCATION*.DESCRIP [DSC$W_LENGTH]) + .DESCRIP [DSC$A_A0];
;  2195          2354  2 !+
;  2196          2355  2 ! Special handling for virtual arrays.
;  2197          2356  2 !-
```

```
;  2198          2357  2
;  2199          2358  2      IF (.DESCRIP [DSC$B_CLASS] EQLU DSC$K_CLASS_BFA)
;  2200          2359  2      THEN
;  2201          2360  2          BEGIN
;  2202          2361  2          BAS$$VA_STORE (.DESCRIP, .VALUE_LOCATION, VALUE);
;  2203          2362  2          END
;  2204          2363  2      ELSE
;  2205          2364  2          BEGIN
;  2206          2365  2
;  2207          2366  2          IF (.DESCRIP [DSC$B_CLASS] NEQU DSC$K_CLASS_A) THEN BAS$$STOP (BAS$K_NOTIMP);
;  2208          2367  2
;  2209          2368  2      !+
;  2210          2369  2      ! Store the value provided into the array
;  2211          2370  2      !-
;  2212          2371  2          BAS$$COPY_F_R1 (VALUE, .VALUE_LOCATION);
;  2213          2372  2          END;
;  2214          2373  2
;  2215          2374  1      END;                                        ! end of BAS$STO_FA_F_R8


                                5E            10  C2 00000 BAS$STO_FA_F_R8::
                                                            SUBL2   #16, SP                                 ; 2243
                                55            51  D0 00003          MOVL    R1, R5                         ;
                             0C AE            50  D0 00006          MOVL    R0, VALUE                      ;
                                54       08   A5  9A 0000A          MOVZBL  11(DESCRIP), R4                ; 2297
                                                05  13 0000E          BEQL    1$                             ;
                                02            54  91 00010          CMPB    R4, #2                         ;
                                                0B  1B 00013          BLEQU   2$                             ;
                                7E       00G  8F  9A 00015 1$:       MOVZBL  #BAS$K_ONEOR_TWO, -(SP)        ;
                         00000000G       00   01  FB 00019          CALLS   #1, BAS$$STOP                  ;
                                0A       02   A5  91 00020 2$:       CMPB    2(DESCRIP), #10                ; 2303
                                                0B  13 00024          BEQL    3$                             ;
                                7E       00G  8F  9A 00026          MOVZBL  #BAS$K_ARGDONMAT, -(SP)        ;
                         00000000G       00   01  FB 0002A          CALLS   #1, BAS$$STOP                  ;
                             05    0A   A5          06  E1 00031 3$:       BBC     #6, 10(DESCRIP), 4$    ; 2309
                                   0A   A5          95  00036          TSTB    10(DESCRIP)                    ; 2310
                                                0B  19 00039          BLSS    5$                             ;
                                7E       00G  8F  9A 0003B 4$:       MOVZBL  #BAS$K_ARGDONMAT, -(SP)        ; 2312
                         00000000G       00   01  FB 0003F          CALLS   #1, BAS$$STOP                  ;
                                04       AE   14  A5  9E 00046 5$:       MOVAB   20(R5), MULTIPLIERS            ; 2314
                                56       14 A544  DE 0004B          MOVAL   20(DESCRIP)[R4], BOUNDS        ; 2315
                             0B    0A   A5          05  E1 00050          BBC     #5, 10(DESCRIP), 6$    ; 2321
                                                51  54  D0 00055          MOVL    R4, LOW_INDEX          ; 2324
                                                50  01  D0 00058          MOVL    #1, HIGH_INDEX         ; 2325
                                                6E  01  CE 0005B          MNEGL   #1, INDEX_INCR         ; 2326
                                                    04  11 0005E          BRB     7$                     ; 2321
                                                51  01  D0 00060 6$:       MOVL    #1, LOW_INDEX          ; 2330
                                                50  54  D0 00063          MOVL    R4, HIGH_INDEX         ; 2331
                                                6E  01  D0 00066          MOVL    #1, INDEX_INCR         ; 2332
                             54                 51  6E  C3 00069 7$:       SUBL3   INDEX_INCR, LOW_INDEX, INDEX_NUMBER ; 2335
                                                57  D4 0006D          CLRL    VALUE_LOCATION         ; 2339
                          08 AE                 50  6E  C1 0006F          ADDL3   INDEX_INCR, HIGH_INDEX, 8(SP) ; 2341
                                                54  6E  C0 00074 8$:       ADDL2   INDEX_INCR, INDEX_NUMBER       ;
                                08   AE               54  D1 00077          CMPL    INDEX_NUMBER, 8(SP)            ;
```

```
                                   3A  13 0007B         BEQL    13$                                          ;
                   01              54  D1 0007D         CMPL    INDEX_NUMBER, #1                             ; 2343
                                   05  12 00080         BNEQ    9$                                           ;
                   58              52  D0 00082         MOVL    INDEX1, INDEX_VALUE                          ;
                                   03  11 00085         BRB     10$                                          ;
                   58              53  D0 00087 9$:     MOVL    INDEX2, INDEX_VALUE                          ;
     50            54              01  78 0008A 10$:    ASHL    #1, INDEX_NUMBER, R0                         ; 2345
          F8 A640                  58  D1 0008E         CMPL    INDEX_VALUE, -8(BOUNDS)[R0]                  ;
                                   07  19 00093         BLSS    11$                                          ;
          FC A640                  58  D1 00095         CMPL    INDEX_VALUE, -4(BOUNDS)[R0]                  ; 2346
                                   0B  15 0009A         BLEQ    12$                                          ;
                   7E        00G   8F  9A 0009C 11$:    MOVZBL  #BAS$K_SUBOUTRAN, -(SP)                      ; 2348
       00000000G   00              01  FB 000A0         CALLS   #1, BAS$$STOP                                ;
     50    04      AE              04  C3 000A7 12$:    SUBL3   #4, MULTIPLIERS, R0                          ; 2350
     51            57            6044  C5 000AC         MULL3   (R0)[INDEX_NUMBER], VALUE_LOCATION, R1       ;
     57            51              58  C1 000B1         ADDL3   INDEX_VALUE, R1, VALUE_LOCATION              ;
                                   BD  11 000B5         BRB     8$                                           ; 2341
                   51              65  3C 000B7 13$:    MOVZWL  (DESCRIP), R1                                ; 2353
                   51              57  C4 000BA         MULL2   VALUE_LOCATION, R1                           ;
     57            51        10    A5  C1 000BD         ADDL3   16(DESCRIP), R1, VALUE_LOCATION              ;
             BF    8F        03    A5  91 000C2         CMPB    3(DESCRIP), #191                             ; 2358
                                   10  12 000C7         BNEQ    14$                                          ;
                             0C    AE  9F 000C9         PUSHAB  VALUE                                        ; 2361
                           00A0    8F  BB 000CC         PUSHR   #^M<R5,R7>                                   ;
       00000000G   00              03  FB 000D0         CALLS   #3, BAS$$VA_STORE                            ;
                                   1E  11 000D7         BRB     16$                                          ; 2358
                   04        03    A5  91 000D9 14$:    CMPB    3(DESCRIP), #4                               ; 2366
                                   0B  13 000DD         BEQL    15$                                          ;
                   7E        00G   8F  9A 000DF         MOVZBL  #BAS$K_NOTIMP, -(SP)                         ;
       00000000G   00              01  FB 000E3         CALLS   #1, BAS$$STOP                                ;
                   50        0C    AE  9E 000EA 15$:    MOVAB   VALUE, R0                                    ; 2371
                   51              57  D0 000EE         MOVL    VALUE_LOCATION, R1                           ;
                     00000000G     00  16 000F1         JSB     BAS$$COPY_F_R1                               ;
                   5E              10  C0 000F7 16$:    ADDL2   #16, SP                                      ; 2374
                                       05 000FA         RSB                                                  ;

; Routine Size:  251 bytes,    Routine Base:  _BAS$CODE + 0F19

; 2216          2375  1
```

```
;  2218          2376  1 GLOBAL ROUTINE BAS$FET_FA_D_R8 (                      ! Fetch a double-floating
;  2219          2377  1         DESCRIP : REF BLOCK [8, BYTE],                ! The descriptor to fetch from
;  2220          2378  1         INDEX1,                                       ! First index
;  2221          2379  1         INDEX2                                        ! Second index
;  2222          2380  1     ) : VA_JSB NOVALUE =
;  2223          2381  1
;  2224          2382  1 !++
;  2225          2383  1 ! FUNCTIONAL DESCRIPTION:
;  2226          2384  1 !
;  2227          2385  1 !       Fetch a double-floating number from an array or virtual array.
;  2228          2386  1 !
;  2229          2387  1 ! FORMAL PARAMETERS:
;  2230          2388  1 !
;  2231          2389  1 !       DESCRIP.rd.da   The descriptor of the array or virtual array
;  2232          2390  1 !       INDEX1.rl.v     The first index into the array
;  2233          2391  1 !       INDEX2.rl.v     The second index into the array
;  2234          2392  1 !
;  2235          2393  1 ! IMPLICIT INPUTS:
;  2236          2394  1 !
;  2237          2395  1 !       NONE
;  2238          2396  1 !
;  2239          2397  1 ! IMPLICIT OUTPUTS:
;  2240          2398  1 !
;  2241          2399  1 !       NONE
;  2242          2400  1 !
;  2243          2401  1 ! ROUTINE VALUE:
;  2244          2402  1 ! COMPLETION CODES:
;  2245          2403  1 !
;  2246          2404  1 !       The double floating number from the array or virtual array
;  2247          2405  1 !
;  2248          2406  1 ! SIDE EFFECTS:
;  2249          2407  1 !
;  2250          2408  1 !       Signals if an error is encountered.
;  2251          2409  1 !
;  2252          2410  1 !--
;  2253          2411  1
;  2254          2412  2     BEGIN
;  2255          2413  2
;  2256          2414  2     LOCAL
;  2257          2415  2         BOUNDS : REF VECTOR,
;  2258          2416  2         MULTIPLIERS : REF VECTOR,
;  2259          2417  2         LOW_INDEX,
;  2260          2418  2         HIGH_INDEX,
;  2261          2419  2         INDEX_INCR,
;  2262          2420  2         VALUE_LOCATION,
;  2263          2421  2         INDEX_VALUE,
;  2264          2422  2         INDEX_NUMBER,
;  2265          2423  2         VALUE : VECTOR [2],
;  2266          2424  2         TEMP_DESCRIP: REF BLOCK[,BYTE];
;  2267          2425  2
;  2268          2426  2 !+
;  2269          2427  2 ! Be sure the array has at least one but no more than two dimensions.
;  2270          2428  2 !-
;  2271          2429  2
;  2272          2430  2     IF ((.DESCRIP [DSC$B_DIMCT] LSSU 1) OR (.DESCRIP [DSC$B_DIMCT] GTRU 2)) THEN BAS$$STOP (BAS$K_ONEOR_TWO)
;  2273          2431  2
;  2274          2432  2 !+
```

```
;  2275     2433  2 ! Be sure this array or virtual array holds double-floating numbers.
;  2276     2434  2 !-
;  2277     2435  2
;  2278     2436  3     IF (.DESCRIP [DSC$B_DTYPE] NEQU DSC$K_DTYPE_D)
;  2279     2437  2     THEN
;  2280     2438  3         IF (.DESCRIP [DSC$B_DTYPE] EQLU DSC$K_DTYPE_DSC)
;  2281     2439  3         THEN
;  2282     2440  3             !+
;  2283     2441  3             ! Special handling for dynamically mapped arrays.
;  2284     2442  3             !-
;  2285     2443  3             BEGIN
;  2286     2444  3
;  2287     2445  3             TEMP_DESCRIP = .DESCRIP [DSC$A_POINTER];
;  2288     2446  4             IF (.TEMP_DESCRIP [DSC$B_DTYPE] NEQU DSC$K_DTYPE_D)
;  2289     2447  3             THEN
;  2290     2448  3                 BAS$$STOP (BAS$K_ARGDONMAT);
;  2291     2449  3
;  2292     2450  3             END
;  2293     2451  3         ELSE
;  2294     2452  3             BAS$$STOP (BAS$K_ARGDONMAT);
;  2295     2453  3
;  2296     2454  2 !+
;  2297     2455  2 ! The coefficients and bounds must be present
;  2298     2456  2 !-
;  2299     2457  2
;  2300     2458  4     IF ( NOT (.DESCRIP [DSC$V_FL_COEFF] AND     !
;  2301     2459  3         .DESCRIP [DSC$V_FL_BOUNDS]))
;  2302     2460  2     THEN
;  2303     2461  2         BAS$$STOP (BAS$K_ARGDONMAT);
;  2304     2462  2
;  2305     2463  2     MULTIPLIERS = DESCRIP [DSC$L_M1];
;  2306     2464  2     BOUNDS = DESCRIP [DSC$L_M1] + (%UPVAL*.DESCRIP [DSC$B_DIMCT]);
;  2307     2465  2 !+
;  2308     2466  2 ! Compute the lower and upper index numbers based on how the array
;  2309     2467  2 ! is stored.
;  2310     2468  2 !-
;  2311     2469  2
;  2312     2470  2     IF (.DESCRIP [DSC$V_FL_COLUMN])
;  2313     2471  2     THEN
;  2314     2472  2         BEGIN
;  2315     2473  2         LOW_INDEX = .DESCRIP [DSC$B_DIMCT];
;  2316     2474  2         HIGH_INDEX = 1;
;  2317     2475  2         INDEX_INCR = -1;
;  2318     2476  2         END
;  2319     2477  2     ELSE
;  2320     2478  2         BEGIN
;  2321     2479  2         LOW_INDEX = 1;
;  2322     2480  2         HIGH_INDEX = .DESCRIP [DSC$B_DIMCT];
;  2323     2481  2         INDEX_INCR = 1;
;  2324     2482  2         END;
;  2325     2483  2
;  2326     2484  2     INDEX_NUMBER = .LOW_INDEX - .INDEX_INCR;
;  2327     2485  2 !+
;  2328     2486  2 ! Compute the linear index from the indices provided.
;  2329     2487  2 !-
;  2330     2488  2     VALUE_LOCATION = 0;
;  2331     2489  2
```

```
;  2332     2490  2     WHILE ((INDEX_NUMBER = .INDEX_NUMBER + .INDEX_INCR) NEQ (.HIGH_INDEX + .INDEX_INCR)) DO
;  2333     2491  3         BEGIN
;  2334     2492  3         INDEX_VALUE = (IF (.INDEX_NUMBER EQL 1) THEN .INDEX1 ELSE .INDEX2);
;  2335     2493  3
;  2336     2494  3         IF ((.INDEX_VALUE LSS .BOUNDS [(.INDEX_NUMBER - 1)*2])  !
;  2337     2495  4             OR (.INDEX_VALUE GTR .BOUNDS [((.INDEX_NUMBER - 1)*2) + 1]))
;  2338     2496  3         THEN
;  2339     2497  4             BAS$$STOP (BAS$K_SUBOUTRAN);
;  2340     2498  3
;  2341     2499  3         VALUE_LOCATION = (.VALUE_LOCATION+.MULTIPLIERS [.INDEX_NUMBER - 1]) + .INDEX_VALUE;
;  2342     2500  3         END;
;  2343     2501  2
;  2344     2502  2     VALUE_LOCATION = (.VALUE_LOCATION+.DESCRIP [DSC$W_LENGTH]) + .DESCRIP [DSC$A_A0];
;  2345     2503  2     IF .DESCRIP [DSC$B_DTYPE] EQLU DSC$K_DTYPE_DSC
;  2346     2504  2     THEN
;  2347     2505  3         !+
;  2348     2506  3         ! Special handling for dynamically mapped arrays.
;  2349     2507  3         !-
;  2350     2508  3         BEGIN
;  2351     2509  3
;  2352     2510  3         TEMP_DESCRIP = .VALUE_LOCATION;
;  2353     2511  3         VALUE_LOCATION = .TEMP_DESCRIP [DSC$A_POINTER];
;  2354     2512  3
;  2355     2513  3         END;
;  2356     2514  2 !+
;  2357     2515  2 ! Special handling for virtual arrays.
;  2358     2516  2 !-
;  2359     2517  2
;  2360     2518  3     IF (.DESCRIP [DSC$B_CLASS] EQLU DSC$K_CLASS_BFA)
;  2361     2519  2     THEN
;  2362     2520  3         BEGIN
;  2363     2521  3         BAS$$VA_FETCH (.DESCRIP, .VALUE_LOCATION, VALUE);
;  2364     2522  4         BEGIN
;  2365     2523  4
;  2366     2524  4         REGISTER
;  2367     2525  4             R0 = 0,
;  2368     2526  4             R1 = 1;
;  2369     2527  4
;  2370     2528  4         R0 = .VALUE [0];
;  2371     2529  4         R1 = .VALUE [1];
;  2372     2530  4         RETURN;
;  2373     2531  3         END;
;  2374     2532  2         END;
;  2375     2533  2
;  2376     2534  2     IF (.DESCRIP [DSC$B_CLASS] NEQU DSC$K_CLASS_A) THEN BAS$$STOP (BAS$K_NOTIMP);
;  2377     2535  2
;  2378     2536  2 !+
;  2379     2537  2 ! Return the array element as our value.
;  2380     2538  2 !-
;  2381     2539  2     BEGIN
;  2382     2540  2
;  2383     2541  3     REGISTER
;  2384     2542  3         R0 = 0,
;  2385     2543  3         R1 = 1;
;  2386     2544  3
;  2387     2545  3     BAS$$COPY_D_R1 (.VALUE_LOCATION, VALUE);
;  2388     2546  3     R0 = .VALUE [0];
```

```
;  2389          2547  3     R1 = .VALUE [1];
;  2390          2548  3     RETURN;
;  2391          2549  2     END;
;  2392          2550  1     END;                                    ! end of BAS$FET_FA_D_R8


                                                                    .TITLE  BAS$VIRTUAL_ARR
                                                                    .IDENT  'J 4'
                                                                    .EXTRN  BAS$$STOP
                                                                    .EXTRN  BAS$K_ARGDONMAT
                                                                    .EXTRN  BAS$K_ONEOR_TWO
                                                                    .PSECT  BAS$CODE,NOWRT,2

                                                                    .ENTRY  BAS$FET_FA_D_R8, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                                                    SUBL2   #20, SP                                         ; 2376
                                                                    PUSHL   R1
                                                                    MOVL    R0, R4
                                                                    MOVZBL  11(DESCRIP), R3                                 ; 2430
                                                                    BEQL    1$
                                                                    CMPB    R3, #2
                                                                    BLEQU   2$
                                                                    MOVZBL  #BAS$K_ONEOR_TWO, -(SP)
                                                                    CALLS   #1, BAS$$STOP
                                                                    CMPB    2(DESCRIP), #11                                 ; 2436
                                                                    BEQL    4$
                                                                    CMPB    2(DESCRIP), #24                                 ; 2438
                                                                    BNEQ    3$
                                                                    MOVL    4(DESCRIP), TEMP_DESCRIP                        ; 2445
                                                                    CMPB    2(TEMP_DESCRIP), #11                            ; 2446
                                                                    BEQL    4$
                                                                    MOVZBL  #BAS$K_ARGDONMAT, -(SP)                         ; 2452
                                                                    CALLS   #1, BAS$$STOP
                                                                    BBC     #6, 10(DESCRIP), 5$                             ; 2458
                                                                    TSTB    10(DESCRIP)                                     ; 2459
                                                                    BLSS    6$
                                                                    MOVZBL  #BAS$K_ARGDONMAT, -(SP)                         ; 2461
                                                                    CALLS   #1, BAS$$STOP
                                                                    MOVAB   20(R4), MULTIPLIERS                             ; 2463
                                                                    MOVAL   20(DESCRIP)[R3], BOUNDS                         ; 2464
                                                                    BBC     #5, 10(DESCRIP), 7$                             ; 2470
                                                                    MOVL    R3, LOW_INDEX                                   ; 2473
                                                                    MOVL    #1, HIGH_INDEX                                  ; 2474
                                                                    MNEGL   #1, INDEX_INCR                                  ; 2475
                                                                    BRB     8$                                              ; 2470
                                                                    MOVL    #1, LOW_INDEX                                   ; 2479
                                                                    MOVL    R3, HIGH_INDEX                                  ; 2480
                                                                    MOVL    #1, INDEX_INCR                                  ; 2481
                                                                    SUBL3   INDEX_INCR, LOW_INDEX, INDEX_NUMBER             ; 2484
                                                                    CLRL    VALUE_LOCATION                                  ; 2488
                                                                    MOVAB   @INDEX_INCR[HIGH_INDEX], 12(SP)                 ; 2490
                                                                    ADDL2   INDEX_INCR, INDEX_NUMBER
                                                                    CMPL    INDEX_NUMBER, 12(SP)
                                                                    BEQL    14$
                                                                    CMPL    INDEX_NUMBER, #1                                ; 2492
                                                                    BNEQ    10$
                                                                    MOVL    INDEX1, INDEX_VALUE
                                                                    BRB     11$
                                                                    MOVL    INDEX2, INDEX_VALUE
                                                                    ASHL    #1, INDEX_NUMBER, R0                            ; 2494
                                                                    CMPL    INDEX_VALUE, -8(BOUNDS)[R0]
                                                                    BLSS    12$
```

```
                FC A540              58  D1 000AB          CMPL    INDEX_VALUE, -4(BOUNDS)[R0]                    ; 2495
                                     0B  15 000AD          BLEQ    13$
                      7E       00G   8F  9A 000AF 12$:     MOVZBL  #BAS$K_SUBOUTRAN, -(SP)                        ; 2497
          00000000G   00             01  FB 000B3          CALLS   #1, BAS$$STOP
51              08   AE             04  C3 000BA 13$:     SUBL3   #4, MULTIPLIERS, R1                            ; 2499
50                   57         6143  C5 000BF          MULL3   (R1)[INDEX_NUMBER], VALUE_LOCATION, R0
57                   50             58  C1 000C4          ADDL3   INDEX_VALUE, R0, VALUE_LOCATION
                                     BC  11 000C8          BRB     9$                                             ; 2490
                     50             64  3C 000CA 14$:     MOVZWL  (DESCRIP), R0                                  ; 2502
                     50             57  C4 000CD          MULL2   VALUE_LOCATION, R0
57                   50         10  A4  C1 000D0          ADDL3   16(DESCRIP), R0, VALUE_LOCATION
                     18         02  A4  91 000D5          CMPB    2(DESCRIP), #24                                ; 2503
                                     07  12 000D9          BNEQ    15$
                     56             57  D0 000DB          MOVL    VALUE_LOCATION, TEMP_DESCRIP                   ; 2510
                     57         04  A6  D0 000DE          MOVL    4(TEMP_DESCRIP), VALUE_LOCATION                ; 2511
                BF   8F         03  A4  91 000E2 15$:     CMPB    3(DESCRIP), #191                               ; 2518
                                     10  12 000E7          BNEQ    16$
                         10          AE  9F 000E9          PUSHAB  VALUE                                          ; 2521
                       0090          8F  BB 000EC          PUSHR   #^M<R4,R7>
          00000000G   00             03  FB 000F0          CALLS   #3, BAS$$VA_FETCH
                                     1E  11 000F7          BRB     18$                                            ; 2528
                     04          03  A4  91 000F9 16$:     CMPB    3(DESCRIP), #4                                 ; 2534
                                     0B  13 000FD          BEQL    17$
                      7E       00G   8F  9A 000FF          MOVZBL  #BAS$K_NOTIMP, -(SP)
          00000000G   00             01  FB 00103          CALLS   #1, BAS$$STOP
                     51          10  AE  9E 0010A 17$:     MOVAB   VALUE, R1                                      ; 2545
                     50             57  D0 0010E          MOVL    VALUE_LOCATION, R0
                           00000000G 00  16 00111          JSB     BAS$$COPY_D_R1
                     50          10  AE  7D 00117 18$:     MOVQ    VALUE, R0                                      ; 2546
                     5E             18  C0 0011B          ADDL2   #24, SP                                        ; 2550
                                        05 0011E          RSB                                                    ;

; Routine Size:  287 bytes,    Routine Base:  _BAS$CODE + 1014

; 2393          2551  1
```

```
;  2395          2552  1 GLOBAL ROUTINE BAS$STO_FA_D_R8 (                      ! Store a double-floating value
;  2396          2553  1         VALUE0,                                       ! The value to store
;  2397          2554  1         VALUE1,
;  2398          2555  1         DESCRIP : REF BLOCK [8, BYTE],                ! The descriptor to store into
;  2399          2556  1         INDEX1,                                       ! First index
;  2400          2557  1         INDEX2                                        ! Second index
;  2401          2558  1     ) : VA_JSB NOVALUE =
;  2402          2559  1
;  2403          2560  1 !++
;  2404          2561  1 ! FUNCTIONAL DESCRIPTION:
;  2405          2562  1 !
;  2406          2563  1 !       Store a 64-bit double floating value in an array or virtual array.
;  2407          2564  1 !
;  2408          2565  1 ! FORMAL PARAMETERS:
;  2409          2566  1 !
;  2410          2567  1 !       VALUE.rd.v      The value to store
;  2411          2568  1 !                       (Passed as two longwords: VALUE0 and VALUE1)
;  2412          2569  1 !       DESCRIP.rd.da   The descriptor of the array or virtual array
;  2413          2570  1 !       INDEX1.rl.v     The first index into the array
;  2414          2571  1 !       INDEX2.rl.v     The second index into the array
;  2415          2572  1 !
;  2416          2573  1 ! IMPLICIT INPUTS:
;  2417          2574  1 !
;  2418          2575  1 !       NONE
;  2419          2576  1 !
;  2420          2577  1 ! IMPLICIT OUTPUTS:
;  2421          2578  1 !
;  2422          2579  1 !       NONE
;  2423          2580  1 !
;  2424          2581  1 ! ROUTINE VALUE:
;  2425          2582  1 ! COMPLETION CODES:
;  2426          2583  1 !
;  2427          2584  1 !       NONE
;  2428          2585  1 !
;  2429          2586  1 ! SIDE EFFECTS:
;  2430          2587  1 !
;  2431          2588  1 !       Signals if an error is encountered.
;  2432          2589  1 !
;  2433          2590  1 !--
;  2434          2591  1
;  2435          2592  2     BEGIN
;  2436          2593  2
;  2437          2594  2     LOCAL
;  2438          2595  2         BOUNDS : REF VECTOR,
;  2439          2596  2         MULTIPLIERS : REF VECTOR,
;  2440          2597  2         LOW_INDEX,
;  2441          2598  2         HIGH_INDEX,
;  2442          2599  2         INDEX_INCR,
;  2443          2600  2         VALUE_LOCATION,
;  2444          2601  2         INDEX_VALUE,
;  2445          2602  2         INDEX_NUMBER,
;  2446          2603  2         VALUE : VECTOR [2];
;  2447          2604  2
;  2448          2605  2 !+
;  2449          2606  2 ! Put the double-precision input value into a local where it will be
;  2450          2607  2 ! safe.
;  2451          2608  2 !-
```

```
;  2452          2609  2      VALUE [0] = .VALUE0;
;  2453          2610  2      VALUE [1] = .VALUE1;
;  2454          2611  2  !+
;  2455          2612  2  ! Be sure the array has at least one but no more than two dimensions.
;  2456          2613  2  !-
;  2457          2614  2
;  2458          2615  2      IF ((.DESCRIP [DSC$B_DIMCT] LSSU 1) OR (.DESCRIP [DSC$B_DIMCT] GTRU 2)) THEN BAS$$STOP (BAS$K_ONEOR_TWO)
;  2459          2616  2
;  2460          2617  2  !+
;  2461          2618  2  ! Be sure this array or virtual array holds double-floating values.
;  2462          2619  2  !-
;  2463          2620  2
;  2464          2621  2      IF (.DESCRIP [DSC$B_DTYPE] NEQU DSC$K_DTYPE_D) THEN BAS$$STOP (BAS$K_ARGDONMAT);
;  2465          2622  2
;  2466          2623  2  !+
;  2467          2624  2  ! The coefficients and bounds must be present
;  2468          2625  2  !-
;  2469          2626  2
;  2470          2627  4      IF ( NOT (.DESCRIP [DSC$V_FL_COEFF] AND     !
;  2471          2628  3          .DESCRIP [DSC$V_FL_BOUNDS]))
;  2472          2629  2      THEN
;  2473          2630  3          BAS$$STOP (BAS$K_ARGDONMAT);
;  2474          2631  2
;  2475          2632  2      MULTIPLIERS = DESCRIP [DSC$L_M1];
;  2476          2633  2      BOUNDS = DESCRIP [DSC$L_M1] + (%UPVAL*.DESCRIP [DSC$B_DIMCT]);
;  2477          2634  2  !+
;  2478          2635  2  ! Compute the lower and upper index numbers based on how the array
;  2479          2636  2  ! is stored.
;  2480          2637  2  !-
;  2481          2638  2
;  2482          2639  3      IF (.DESCRIP [DSC$V_FL_COLUMN])
;  2483          2640  3      THEN
;  2484          2641  3          BEGIN
;  2485          2642  3          LOW_INDEX = .DESCRIP [DSC$B_DIMCT];
;  2486          2643  3          HIGH_INDEX = 1;
;  2487          2644  3          INDEX_INCR = -1;
;  2488          2645  3          END
;  2489          2646  3      ELSE
;  2490          2647  3          BEGIN
;  2491          2648  3          LOW_INDEX = 1;
;  2492          2649  3          HIGH_INDEX = .DESCRIP [DSC$B_DIMCT];
;  2493          2650  3          INDEX_INCR = 1;
;  2494          2651  3          END;
;  2495          2652  2
;  2496          2653  2      INDEX_NUMBER = .LOW_INDEX - .INDEX_INCR;
;  2497          2654  2  !+
;  2498          2655  2  ! Compute the linear index from the indices provided.
;  2499          2656  2  !-
;  2500          2657  2      VALUE_LOCATION = 0;
;  2501          2658  2
;  2502          2659  2      WHILE ((INDEX_NUMBER = .INDEX_NUMBER + .INDEX_INCR) NEQ (.HIGH_INDEX + .INDEX_INCR)) DO
;  2503          2660  3          BEGIN
;  2504          2661  3          INDEX_VALUE = (IF (.INDEX_NUMBER EQL 1) THEN .INDEX1 ELSE .INDEX2);
;  2505          2662  3
;  2506          2663  5          IF ((.INDEX_VALUE LSS .BOUNDS [(.INDEX_NUMBER - 1)*2])  !
;  2507          2664  4              OR (.INDEX_VALUE GTR .BOUNDS [((.INDEX_NUMBER - 1)*2) + 1]))
;  2508          2665  3          THEN
```

```
;  2509          2666  3              BAS$$STOP (BAS$K_SUBOUTRAN);
;  2510          2667  3
;  2511          2668  3          VALUE_LOCATION = (.VALUE_LOCATION+.MULTIPLIERS [.INDEX_NUMBER - 1]) + .INDEX_VALUE;
;  2512          2669  3          END;
;  2513          2670  3
;  2514          2671  3      VALUE_LOCATION = (.VALUE_LOCATION+.DESCRIP [DSC$W_LENGTH]) + .DESCRIP [DSC$A_A0];
;  2515          2672  3  !+
;  2516          2673  3  ! Special handling for virtual arrays.
;  2517          2674  3  !-
;  2518          2675  3
;  2519          2676  3      IF (.DESCRIP [DSC$B_CLASS] EQLU DSC$K_CLASS_BFA)
;  2520          2677  3      THEN
;  2521          2678  3          BEGIN
;  2522          2679  3          BAS$$VA_STORE (.DESCRIP, .VALUE_LOCATION, VALUE);
;  2523          2680  3          END
;  2524          2681  3      ELSE
;  2525          2682  3          BEGIN
;  2526          2683  3
;  2527          2684  3          IF (.DESCRIP [DSC$B_CLASS] NEQU DSC$K_CLASS_A) THEN BAS$$STOP (BAS$K_NOTIMP);
;  2528          2685  3
;  2529          2686  3  !+
;  2530          2687  3  ! Store the value provided into the array
;  2531          2688  3  !-
;  2532          2689  3          BAS$$COPY_D_R1 (VALUE [0], .VALUE_LOCATION);
;  2533          2690  3          END;
;  2534          2691  3
;  2535          2692  1      END;                                    ! end of BAS$STO_FA_D_R8


                      5E             14  C2 00000 BAS$STO_FA_D_R8::
                                                   SUBL2   #20, SP                                 ;  2552
                   0C AE             50  7D 00003           MOVQ    VALUE0, VALUE                   ;  2609
                      55          0B A2  9A 00007           MOVZBL  11(DESCRIP), R5                 ;  2615
                                     05  13 0000B           BEQL    1$
                      02                55  91 0000D           CMPB    R5, #2
                                        0B  1B 00010           BLEQU   2$
                      7E             00G  8F  9A 00012 1$:     MOVZBL  #BAS$K_ONEOR_TWO, -(SP)
                00000000G 00             01  FB 00016           CALLS   #1, BAS$$STOP
                      0B             02  A2  91 0001D 2$:     CMPB    2(DESCRIP), #11                 ;  2621
                                        0B  13 00021           BEQL    3$
                      7E             00G  8F  9A 00023           MOVZBL  #BAS$K_ARGDONMAT, -(SP)
                00000000G 00             01  FB 00027           CALLS   #1, BAS$$STOP
                   05    0A AA             06  E1 0002E 3$:     BBC     #6, 10(DESCRIP), 4$             ;  2627
                                     0A  A2  95 00033           TSTB    10(DESCRIP)                     ;  2628
                                        0B  19 00036           BLSS    5$
                      7E             00G  8F  9A 00038 4$:     MOVZBL  #BAS$K_ARGDONMAT, -(SP)         ;  2630
                00000000G 00             01  FB 0003C           CALLS   #1, BAS$$STOP
                      04             AE  14  A2  9E 00043 5$:     MOVAB   20(R2), MULTIPLIERS             ;  2632
                      56          14 A245  DE 00048           MOVAL   20(DESCRIP)[R5], BOUNDS         ;  2633
                   0B    0A AA             05  E1 0004D           BBC     #5, 10(DESCRIP), 6$             ;  2639
                      51                55  D0 00052           MOVL    R5, LOW_INDEX                   ;  2642
                      50                01  D0 00055           MOVL    #1, HIGH_INDEX                  ;  2643
                      6E                01  CE 00058           MNEGL   #1, INDEX_INCR                  ;  2644
                                        09  11 0005B           BRB     7$                              ;  2639
```

```
                        51              01  D0 0005D 6$:    MOVL    #1, LOW_INDEX                           ; 2648
                        50              55  D0 00060        MOVL    R5, HIGH_INDEX                          ; 2649
                        6E              01  D0 00063        MOVL    #1, INDEX_INCR                          ; 2650
                55      51              6E  C3 00066 7$:    SUBL3   INDEX_INCR, LOW_INDEX, INDEX_NUMBER     ; 2653
                                        57  D4 0006A        CLRL    VALUE_LOCATION                          ; 2657
            08  AE      50              6E  C1 0006C        ADDL3   INDEX_INCR, HIGH_INDEX, 8(SP)           ; 2659
                        55              6E  C0 00071 8$:    ADDL2   INDEX_INCR, INDEX_NUMBER
                    08  AE              55  D1 00074        CMPL    INDEX_NUMBER, 8(SP)
                                        3A  13 00078        BEQL    13$
                        01              55  D1 0007A        CMPL    INDEX_NUMBER, #1                        ; 2661
                                        05  12 0007D        BNEQ    9$
                        58              53  D0 0007F        MOVL    INDEX1, INDEX_VALUE
                                        03  11 00082        BRB     10$
                        58              54  D0 00084 9$:    MOVL    INDEX2, INDEX_VALUE
                50      55              01  78 00087 10$:   ASHL    #1, INDEX_NUMBER, R0                    ; 2663
                    F8 A640             58  D1 0008B        CMPL    INDEX_VALUE, -8(BOUNDS)[R0]
                                        07  19 00090        BLSS    11$
                    FC A640             58  D1 00092        CMPL    INDEX_VALUE, -4(BOUNDS)[R0]             ; 2664
                                        0B  15 00097        BLEQ    12$
                        7E      00G     8F  9A 00099 11$:   MOVZBL  #BAS$K_SUBOUTRAN, -(SP)                 ; 2666
            00000000G   00              01  FB 0009D        CALLS   #1, BAS$$STOP
                50      04      AE      04  C3 000A4 12$:   SUBL3   #4, MULTIPLIERS, R0                     ; 2668
                51              57  6045  C5 000A9        MULL3   (R0)[INDEX_NUMBER], VALUE_LOCATION, R1
                57              51      58  C1 000AE        ADDL3   INDEX_VALUE, R1, VALUE_LOCATION
                                        BD  11 000B2        BRB     8$                                      ; 2659
                        51              62  3C 000B4 13$:   MOVZWL  (DESCRIP), R1                          ; 2671
                        51              57  C4 000B7        MULL2   VALUE_LOCATION, R1
                57      51      10      A2  C1 000BA        ADDL3   16(DESCRIP), R1, VALUE_LOCATION
                    BF  8F      03      A2  91 000BF        CMPB    3(DESCRIP), #191                        ; 2676
                                        10  12 000C4        BNEQ    14$
                                    0C  AE  9F 000C6        PUSHAB  VALUE                                   ; 2679
                                0084    8F  BB 000C9        PUSHR   #^M<R2,R7>
                    00000000G   00          03  FB 000CD        CALLS   #3, BAS$$VA_STORE
                                            1E  11 000D4        BRB     16$                                 ; 2676
                        04          03      A2  91 000D6 14$:   CMPB    3(DESCRIP), #4                      ; 2684
                                            0B  13 000DA        BEQL    15$
                        7E          00G     8F  9A 000DC        MOVZBL  #BAS$K_NOTIMP, -(SP)
                    00000000G   00          01  FB 000E0        CALLS   #1, BAS$$STOP
                        50          0C      AE  9E 000E7 15$:   MOVAB   VALUE, R0                           ; 2689
                        51              57  D0 000EB        MOVL    VALUE_LOCATION, R1
                                00000000G   00  16 000EE        JSB     BAS$$COPY_D_R1
                        5E                  14  C0 000F4 16$:   ADDL2   #20, SP                            ; 2692
                                            05  000F7        RSB

; Routine Size:  248 bytes,    Routine Base:  _BAS$CODE + 1133
```

```
;  2537          2693  1 GLOBAL ROUTINE BAS$FET_FA_B_R8 (                      ! Fetch a byte
;  2538          2694  1         DESCRIP : REF BLOCK [8, BYTE],                ! The descriptor to fetch from
;  2539          2695  1         INDEX1,                                       ! First index
;  2540          2696  1         INDEX2                                        ! Second index
;  2541          2697  1     ) : VA_JSB =
;  2542          2698  1
;  2543          2699  1 !++
;  2544          2700  1 ! FUNCTIONAL DESCRIPTION:
;  2545          2701  1 !
;  2546          2702  1 !       Fetch a byte from an array or virtual array.
;  2547          2703  1 !
;  2548          2704  1 ! FORMAL PARAMETERS:
;  2549          2705  1 !
;  2550          2706  1 !       DESCRIP.rw.da   The descriptor of the array or virtual array
;  2551          2707  1 !       INDEX1.rl.v     The first index into the array
;  2552          2708  1 !       INDEX2.rl.v     The second index into the array
;  2553          2709  1 !
;  2554          2710  1 ! IMPLICIT INPUTS:
;  2555          2711  1 !
;  2556          2712  1 !       NONE
;  2557          2713  1 !
;  2558          2714  1 ! IMPLICIT OUTPUTS:
;  2559          2715  1 !
;  2560          2716  1 !       NONE
;  2561          2717  1 !
;  2562          2718  1 ! ROUTINE VALUE:
;  2563          2719  1 ! COMPLETION CODES:
;  2564          2720  1 !
;  2565          2721  1 !       The byte from the array or virtual array
;  2566          2722  1 !
;  2567          2723  1 ! SIDE EFFECTS:
;  2568          2724  1 !
;  2569          2725  1 !       Signals if an error is encountered.
;  2570          2726  1 !
;  2571          2727  1 !--
;  2572          2728  1
;  2573          2729  2     BEGIN
;  2574          2730  2
;  2575          2731  2     LOCAL
;  2576          2732  2         BOUNDS : REF VECTOR,
;  2577          2733  2         MULTIPLIERS : REF VECTOR,
;  2578          2734  2         LOW_INDEX,
;  2579          2735  2         HIGH_INDEX,
;  2580          2736  2         INDEX_INCR,
;  2581          2737  2         VALUE_LOCATION,
;  2582          2738  2         INDEX_VALUE,
;  2583          2739  2         INDEX_NUMBER,
;  2584          2740  2         TEMP_DESCRIP: REF BLOCK[,BYTE];
;  2585          2741  2
;  2586          2742  2 !+
;  2587          2743  2 ! Be sure the array has at least one but no more than two dimensions.
;  2588          2744  2 !-
;  2589          2745  2
;  2590          2746  2     IF ((.DESCRIP [DSC$B_DIMCT] LSSU 1) OR (.DESCRIP [DSC$B_DIMCT] GTRU 2)) THEN BAS$$STOP (BAS$K_ONEOR_TWO)
;  2591          2747  2
;  2592          2748  2 !+
;  2593          2749  2 ! Be sure this array or virtual array holds bytes.
```

```
;  2594          2750  2 !-
;  2595          2751  2
;  2596          2752  2     IF (.DESCRIP [DSC$B_DTYPE] NEQU DSC$K_DTYPE_B)
;  2597          2753  2     THEN
;  2598          2754  2         IF (.DESCRIP [DSC$B_DTYPE] EQLU DSC$K_DTYPE_DSC)
;  2599          2755  2         THEN
;  2600          2756  2             !+
;  2601          2757  2             ! Special handling for dynamically mapped arrays.
;  2602          2758  2             !-
;  2603          2759  2             BEGIN
;  2604          2760  2
;  2605          2761  2             TEMP_DESCRIP = .DESCRIP [DSC$A_POINTER];
;  2606          2762  2             IF (.TEMP_DESCRIP [DSC$B_DTYPE] NEQU DSC$K_DTYPE_B)
;  2607          2763  2             THEN
;  2608          2764  2                 BAS$$STOP (BAS$K_ARGDONMAT);
;  2609          2765  2
;  2610          2766  2             END
;  2611          2767  2         ELSE
;  2612          2768  2             BAS$$STOP (BAS$K_ARGDONMAT);
;  2613          2769  2
;  2614          2770  2     !+
;  2615          2771  2     ! The coefficients and bounds must be present
;  2616          2772  2     !-
;  2617          2773  2
;  2618          2774  2     IF ( NOT (.DESCRIP [DSC$V_FL_COEFF] AND .DESCRIP [DSC$V_FL_BOUNDS])) THEN BAS$$STOP (BAS$K_ARGDONMAT);
;  2619          2775  2
;  2620          2776  2     MULTIPLIERS = DESCRIP [DSC$L_M1];
;  2621          2777  2     BOUNDS = DESCRIP [DSC$L_M1] + (%UPVAL*.DESCRIP [DSC$B_DIMCT]);
;  2622          2778  2     !+
;  2623          2779  2     ! Compute the lower and upper index numbers based on how the array
;  2624          2780  2     ! is stored.
;  2625          2781  2     !-
;  2626          2782  2
;  2627          2783  2     IF (.DESCRIP [DSC$V_FL_COLUMN])
;  2628          2784  2     THEN
;  2629          2785  2         BEGIN
;  2630          2786  2         LOW_INDEX = .DESCRIP [DSC$B_DIMCT];
;  2631          2787  2         HIGH_INDEX = 1;
;  2632          2788  2         INDEX_INCR = -1;
;  2633          2789  2         END
;  2634          2790  2     ELSE
;  2635          2791  2         BEGIN
;  2636          2792  2         LOW_INDEX = 1;
;  2637          2793  2         HIGH_INDEX = .DESCRIP [DSC$B_DIMCT];
;  2638          2794  2         INDEX_INCR = 1;
;  2639          2795  2         END;
;  2640          2796  2
;  2641          2797  2     INDEX_NUMBER = .LOW_INDEX - .INDEX_INCR;
;  2642          2798  2     !+
;  2643          2799  2     ! Compute the linear index from the indices provided.
;  2644          2800  2     !-
;  2645          2801  2     VALUE_LOCATION = 0;
;  2646          2802  2
;  2647          2803  2     WHILE ((INDEX_NUMBER = .INDEX_NUMBER + .INDEX_INCR) NEQ (.HIGH_INDEX + .INDEX_INCR)) DO
;  2648          2804  2         BEGIN
;  2649          2805  2         INDEX_VALUE = (IF (.INDEX_NUMBER EQL 1) THEN .INDEX1 ELSE .INDEX2);
;  2650          2806  2
```

```
;  2651     2807  5          IF ((.INDEX_VALUE LSS .BOUNDS [(.INDEX_NUMBER - 1)*2])  !
;  2652     2808  4              OR (.INDEX_VALUE GTR .BOUNDS [((.INDEX_NUMBER - 1)*2) + 1]))
;  2653     2809  4          THEN
;  2654     2810  4              BAS$$STOP (BAS$K_SUBOUTRAN);
;  2655     2811  4
;  2656     2812  4          VALUE_LOCATION = (.VALUE_LOCATION+.MULTIPLIERS [.INDEX_NUMBER - 1]) + .INDEX_VALUE;
;  2657     2813  4          END;
;  2658     2814  3
;  2659     2815  3      VALUE_LOCATION = (.VALUE_LOCATION+.DESCRIP [DSC$W_LENGTH]) + .DESCRIP [DSC$A_A0];
;  2660     2816  3      IF .DESCRIP [DSC$B_DTYPE] EQLU DSC$K_DTYPE_DSC
;  2661     2817  3      THEN
;  2662     2818  4          !+
;  2663     2819  4          ! Special handling for dynamically mapped arrays.
;  2664     2820  4          !-
;  2665     2821  4          BEGIN
;  2666     2822  4
;  2667     2823  4          TEMP_DESCRIP = .VALUE_LOCATION;
;  2668     2824  4          VALUE_LOCATION = .TEMP_DESCRIP [DSC$A_POINTER];
;  2669     2825  4
;  2670     2826  4          END;
;  2671     2827  3  !+
;  2672     2828  3  ! Special handling if this is a virtual array.
;  2673     2829  3  !-
;  2674     2830  3
;  2675     2831  3      IF (.DESCRIP [DSC$B_CLASS] EQLU DSC$K_CLASS_BFA)
;  2676     2832  3      THEN
;  2677     2833  4          BEGIN
;  2678     2834  4
;  2679     2835  4          LOCAL
;  2680     2836  4              VALUE;
;  2681     2837  4
;  2682     2838  4          VALUE = 0;
;  2683     2839  4          BAS$$VA_FETCH (.DESCRIP, .VALUE_LOCATION, VALUE);
;  2684     2840  4          RETURN (.VALUE);
;  2685     2841  4          END;
;  2686     2842  3
;  2687     2843  3      IF (.DESCRIP [DSC$B_CLASS] NEQU DSC$K_CLASS_A) THEN BAS$$STOP (BAS$K_NOTIMP);
;  2688     2844  3
;  2689     2845  3  !+
;  2690     2846  3  ! Return the array element as our value.
;  2691     2847  3  !-
;  2692     2848  3      RETURN (.BLOCK [.VALUE_LOCATION, 0, 0, %BPUNIT, 1]);
;  2693     2849  2      END;                                    ! end of BAS$FET_FA_B_R8


                     5E            10  C2 00000 BAS$FET_FA_B_R8::
                                               SUBL2   #16, SP                          ; 2693
                                   51  DD 00003         PUSHL   R1
                     55            50  D0 00005         MOVL    R0, R5
                     53       0B   A5  9A 00008         MOVZBL  11(DESCRIP), R3          ; 2746
                                   05  13 0000C         BEQL    1$
                     02            53  91 0000E         CMPB    R3, #2
                                   0B  1B 00011         BLEQU   2$
                     7E       00G  8F  9A 00013 1$:     MOVZBL  #BAS$K_ONEOR_TWO, -(SP)
```

```
                00000000G  00          01  FB 00017          CALLS   #1, BAS$$STOP                                    ;
                           06      02  A5  91 0001E 2$:      CMPB    2(DESCRIP), #6                                   ; 2752
                                       1B  13 00022          BEQL    4$                                               ;
                           18      02  A5  91 00024          CMPB    2(DESCRIP), #24                                  ; 2754
                                       0A  12 00028          BNEQ    3$                                               ;
                           57      04  A5  D0 0002A          MOVL    4(DESCRIP), TEMP_DESCRIP                         ; 2761
                           06      02  A7  91 0002E          CMPB    2(TEMP_DESCRIP), #6                              ; 2762
                                       0B  13 00032          BEQL    4$                                               ;
                           7E      00G 8F  9A 00034 3$:      MOVZBL  #BAS$K_ARGDONMAT, -(SP)                          ; 2768
                00000000G  00          01  FB 00038          CALLS   #1, BAS$$STOP                                    ;
                     05        0A  A5      06  E1 0003F 4$:  BBC     #6, 10(DESCRIP), 5$                              ; 2774
                                   0A  A5  95 00044          TSTB    10(DESCRIP)                                      ;
                                       0B  19 00047          BLSS    6$                                               ;
                           7E      00G 8F  9A 00049 5$:      MOVZBL  #BAS$K_ARGDONMAT, -(SP)                          ;
                00000000G  00          01  FB 0004D          CALLS   #1, BAS$$STOP                                    ;
                           08      AE  14  A5  9E 00054 6$:  MOVAB   20(R5), MULTIPLIERS                              ; 2776
                           56      14 A543 DE 00059          MOVAL   20(DESCRIP)[R3], BOUNDS                          ; 2777
                     0C        0A  A5      05  E1 0005E      BBC     #5, 10(DESCRIP), 7$                              ; 2783
                           51              53  D0 00063      MOVL    R3, LOW_INDEX                                    ; 2786
                           50              01  D0 00066      MOVL    #1, HIGH_INDEX                                   ; 2787
                                   04  AE  01  CE 00069      MNEGL   #1, INDEX_INCR                                   ; 2788
                                           0A  11 0006D      BRB     8$                                               ; 2783
                           51              01  D0 0006F 7$:  MOVL    #1, LOW_INDEX                                    ; 2792
                           50              53  D0 00072      MOVL    R3, HIGH_INDEX                                   ; 2793
                                   04  AE  01  D0 00075      MOVL    #1, INDEX_INCR                                   ; 2794
                     54            51  04  AE  C3 00079 8$:  SUBL3   INDEX_INCR, LOW_INDEX, INDEX_NUMBER            ; 2797
                                           53  D4 0007E      CLRL    VALUE_LOCATION                                   ; 2801
                                   0C  AE  04 BE40 9E 00080  MOVAB   @INDEX_INCR[HIGH_INDEX], 12(SP)                 ; 2803
                           54      04  AE  C0 00086 9$:      ADDL2   INDEX_INCR, INDEX_NUMBER                         ;
                                   0C  AE      54  D1 0008A  CMPL    INDEX_NUMBER, 12(SP)                             ;
                                               3A  13 0008E  BEQL    14$                                              ;
                           01              54  D1 00090      CMPL    INDEX_NUMBER, #1                                 ; 2805
                                           05  12 00093      BNEQ    10$                                              ;
                           58              6E  D0 00095      MOVL    INDEX1, INDEX_VALUE                              ;
                                           03  11 00098      BRB     11$                                              ;
                           58              52  D0 0009A 10$:  MOVL    INDEX2, INDEX_VALUE                              ;
                     50            54      01  78 0009D 11$:  ASHL    #1, INDEX_NUMBER, R0                             ; 2807
                                 F8 A640       58  D1 000A1  CMPL    INDEX_VALUE, -8(BOUNDS)[R0]                      ;
                                               07  19 000A6  BLSS    12$                                              ;
                                 FC A640       58  D1 000A8  CMPL    INDEX_VALUE, -4(BOUNDS)[R0]                      ; 2808
                                               0B  15 000AD  BLEQ    13$                                              ;
                           7E              00G 8F  9A 000AF 12$:  MOVZBL  #BAS$K_SUBOUTRAN, -(SP)                  ; 2810
                00000000G  00          01  FB 000B3          CALLS   #1, BAS$$STOP                                    ;
                     51        08  AE      04  C3 000BA 13$:  SUBL3   #4, MULTIPLIERS, R1                             ; 2812
                     50              53    6144  C5 000BF      MULL3   (R1)[INDEX_NUMBER], VALUE_LOCATION, R0           ;
                     53              50      58  C1 000C4      ADDL3   INDEX_VALUE, R0, VALUE_LOCATION                 ;
                                             BC  11 000C8      BRB     9$                                               ; 2803
                           50              65  3C 000CA 14$:  MOVZWL  (DESCRIP), R0                                   ; 2815
                           50              53  C4 000CD      MULL2   VALUE_LOCATION, R0                               ;
                     53    50      10  A5  C1 000D0          ADDL3   16(DESCRIP), R0, VALUE_LOCATION                  ;
                           18      02  A5  91 000D5          CMPB    2(DESCRIP), #24                                  ; 2816
                                           07  12 000D9      BNEQ    15$                                              ;
                           57              53  D0 000DB      MOVL    VALUE_LOCATION, TEMP_DESCRIP                     ; 2823
                           53      04  A7  D0 000DE          MOVL    4(TEMP_DESCRIP), VALUE_LOCATION                  ; 2824
                                 BF    8F  03  A5  91 000E2 15$:  CMPB    3(DESCRIP), #191                             ; 2831
                                           17  12 000E7      BNEQ    16$                                              ;
                                       10  AE  D4 000E9      CLRL    VALUE                                            ; 2838
```

```
                                 10  AE  9F 000EC          PUSHAB  VALUE                                   ; 2839
                                     53  DD 000EF          PUSHL   VALUE_LOCATION                          ;
                                     55  DD 000F1          PUSHL   DESCRIP                                 ;
                00000000G  00        03  FB 000F3          CALLS   #3, BAS$$VA_FETCH                       ;
                           50    10  AE  D0 000FA          MOVL    VALUE, R0                               ; 2840
                                     14  11 000FE          BRB     18$                                     ;
                           04    03  A5  91 00100 16$:     CMPB    3(DESCRIP), #4                          ; 2843
                                     0B  13 00104          BEQL    17$                                     ;
                           7E    00G 8F  9A 00106          MOVZBL  #BAS$K_NOTIMP, -(SP)                    ;
                00000000G  00        01  FB 0010A          CALLS   #1, BAS$$STOP                           ;
                           50        63  98 00111 17$:     CVTBL   (VALUE_LOCATION), R0                    ; 2848
                           5E        14  C0 00114 18$:     ADDL2   #20, SP                                 ; 2849
                                         05 00117          RSB                                             ;

; Routine Size:  280 bytes,    Routine Base:  _BAS$CODE + 122B
```

```
;  2695          2850  1 GLOBAL ROUTINE BAS$FET_FA_G_RB (                      ! Fetch a g floating
;  2696          2851  1         DESCRIP : REF BLOCK [8, BYTE],                ! The descriptor to fetch from
;  2697          2852  1         INDEX1,                                       ! First index
;  2698          2853  1         INDEX2                                        ! Second index
;  2699          2854  1     ) : VA_JSB NOVALUE =
;  2700          2855  1
;  2701          2856  1 !++
;  2702          2857  1 ! FUNCTIONAL DESCRIPTION:
;  2703          2858  1 !
;  2704          2859  1 !       Fetch a g floating number from an array or virtual array.
;  2705          2860  1 !
;  2706          2861  1 ! FORMAL PARAMETERS:
;  2707          2862  1 !
;  2708          2863  1 !       DESCRIP.rd.da   The descriptor of the array or virtual array
;  2709          2864  1 !       INDEX1.rl.v     The first index into the array
;  2710          2865  1 !       INDEX2.rl.v     The second index into the array
;  2711          2866  1 !
;  2712          2867  1 ! IMPLICIT INPUTS:
;  2713          2868  1 !
;  2714          2869  1 !       NONE
;  2715          2870  1 !
;  2716          2871  1 ! IMPLICIT OUTPUTS:
;  2717          2872  1 !
;  2718          2873  1 !       NONE
;  2719          2874  1 !
;  2720          2875  1 ! ROUTINE VALUE:
;  2721          2876  1 ! COMPLETION CODES:
;  2722          2877  1 !
;  2723          2878  1 !       The g floating number from the array or virtual array
;  2724          2879  1 !
;  2725          2880  1 ! SIDE EFFECTS:
;  2726          2881  1 !
;  2727          2882  1 !       Signals if an error is encountered.
;  2728          2883  1 !
;  2729          2884  1 !--
;  2730          2885  1
;  2731          2886  2     BEGIN
;  2732          2887  2
;  2733          2888  2     LOCAL
;  2734          2889  2         BOUNDS : REF VECTOR,
;  2735          2890  2         MULTIPLIERS : REF VECTOR,
;  2736          2891  2         LOW_INDEX,
;  2737          2892  2         HIGH_INDEX,
;  2738          2893  2         INDEX_INCR,
;  2739          2894  2         VALUE_LOCATION,
;  2740          2895  2         INDEX_VALUE,
;  2741          2896  2         INDEX_NUMBER,
;  2742          2897  2         VALUE : VECTOR [2],
;  2743          2898  2         TEMP_DESCRIP: REF BLOCK[,BYTE];
;  2744          2899  2
;  2745          2900  2 !+
;  2746          2901  2 ! Be sure the array has at least one but no more than two dimensions.
;  2747          2902  2 !-
;  2748          2903  2
;  2749          2904  2     IF ((.DESCRIP [DSC$B_DIMCT] LSSU 1) OR (.DESCRIP [DSC$B_DIMCT] GTRU 2)) THEN BAS$$STOP (BAS$K_ONEOR_TWO)
;  2750          2905  2
;  2751          2906  2 !+
```

```
;  2752          2907  2 !+ Be sure this array or virtual array holds g floating numbers.
;  2753          2908  2 !-
;  2754          2909  2
;  2755          2910  2     IF (.DESCRIP [DSC$B_DTYPE] NEQU DSC$K_DTYPE_G)
;  2756          2911  2     THEN
;  2757          2912  2         IF (.DESCRIP [DSC$B_DTYPE] EQLU DSC$K_DTYPE_DSC)
;  2758          2913  2         THEN
;  2759          2914  2             !+
;  2760          2915  2             ! Special handling for dynamically mapped arrays.
;  2761          2916  2             !-
;  2762          2917  2             BEGIN
;  2763          2918  2
;  2764          2919  3             TEMP_DESCRIP = .DESCRIP [DSC$A_POINTER];
;  2765          2920  3             IF (.TEMP_DESCRIP [DSC$B_DTYPE] NEQU DSC$K_DTYPE_G)
;  2766          2921  3             THEN
;  2767          2922  3                 BAS$$STOP (BAS$K_ARGDONMAT);
;  2768          2923  3
;  2769          2924  3             END
;  2770          2925  2         ELSE
;  2771          2926  2             BAS$$STOP (BAS$K_ARGDONMAT);
;  2772          2927  2
;  2773          2928  2 !+
;  2774          2929  2 ! The coefficients and bounds must be present
;  2775          2930  2 !-
;  2776          2931  2
;  2777          2932  2     IF ( NOT (.DESCRIP [DSC$V_FL_COEFF] AND     !
;  2778          2933  2         .DESCRIP [DSC$V_FL_BOUNDS]))
;  2779          2934  2     THEN
;  2780          2935  2         BAS$$STOP (BAS$K_ARGDONMAT);
;  2781          2936  2
;  2782          2937  2     MULTIPLIERS = DESCRIP [DSC$L_M1];
;  2783          2938  2     BOUNDS = DESCRIP [DSC$L_M1] + (%UPVAL*.DESCRIP [DSC$B_DIMCT]);
;  2784          2939  2 !+
;  2785          2940  2 ! Compute the lower and upper index numbers based on how the array
;  2786          2941  2 ! is stored.
;  2787          2942  2 !-
;  2788          2943  2
;  2789          2944  2     IF (.DESCRIP [DSC$V_FL_COLUMN])
;  2790          2945  2     THEN
;  2791          2946  3         BEGIN
;  2792          2947  3         LOW_INDEX = .DESCRIP [DSC$B_DIMCT];
;  2793          2948  3         HIGH_INDEX = 1;
;  2794          2949  3         INDEX_INCR = -1;
;  2795          2950  3         END
;  2796          2951  2     ELSE
;  2797          2952  3         BEGIN
;  2798          2953  3         LOW_INDEX = 1;
;  2799          2954  3         HIGH_INDEX = .DESCRIP [DSC$B_DIMCT];
;  2800          2955  3         INDEX_INCR = 1;
;  2801          2956  3         END;
;  2802          2957  2
;  2803          2958  2     INDEX_NUMBER = .LOW_INDEX - .INDEX_INCR;
;  2804          2959  2 !+
;  2805          2960  2 ! Compute the linear index from the indices provided.
;  2806          2961  2 !-
;  2807          2962  2     VALUE_LOCATION = 0;
;  2808          2963  2
```

```
;  2809          2964  2     WHILE ((INDEX_NUMBER = .INDEX_NUMBER + .INDEX_INCR) NEQ (.HIGH_INDEX + .INDEX_INCR)) DO
;  2810          2965  2         BEGIN
;  2811          2966  2         INDEX_VALUE = (IF (.INDEX_NUMBER EQL 1) THEN .INDEX1 ELSE .INDEX2);
;  2812          2967  2
;  2813          2968  2         IF ((.INDEX_VALUE LSS .BOUNDS [(.INDEX_NUMBER - 1)*2])  !
;  2814          2969  2             OR (.INDEX_VALUE GTR .BOUNDS [((.INDEX_NUMBER - 1)*2) + 1]))
;  2815          2970  2         THEN
;  2816          2971  2             BAS$$STOP (BAS$K_SUBOUTRAN);
;  2817          2972  2
;  2818          2973  2         VALUE_LOCATION = (.VALUE_LOCATION+.MULTIPLIERS [.INDEX_NUMBER - 1]) + .INDEX_VALUE;
;  2819          2974  2         END;
;  2820          2975  2
;  2821          2976  2     VALUE_LOCATION = (.VALUE_LOCATION+.DESCRIP [DSC$W_LENGTH]) + .DESCRIP [DSC$A_A0];
;  2822          2977  2     IF .DESCRIP [DSC$B_DTYPE] EQLU DSC$K_DTYPE_DSC
;  2823          2978  2     THEN
;  2824          2979  2         !+
;  2825          2980  2         ! Special handling for dynamically mapped arrays.
;  2826          2981  2         !-
;  2827          2982  2         BEGIN
;  2828          2983  2
;  2829          2984  2         TEMP_DESCRIP = .VALUE_LOCATION;
;  2830          2985  2         VALUE_LOCATION = .TEMP_DESCRIP [DSC$A_POINTER];
;  2831          2986  2
;  2832          2987  2         END;
;  2833          2988  2     !+
;  2834          2989  2     ! Special handling for virtual arrays.
;  2835          2990  2     !-
;  2836          2991  2
;  2837          2992  2     IF (.DESCRIP [DSC$B_CLASS] EQLU DSC$K_CLASS_BFA)
;  2838          2993  2     THEN
;  2839          2994  3         BEGIN
;  2840          2995  3         BAS$$VA_FETCH (.DESCRIP, .VALUE_LOCATION, VALUE);
;  2841          2996  4         BEGIN
;  2842          2997  4
;  2843          2998  4         REGISTER
;  2844          2999  4             R0 = 0,
;  2845          3000  4             R1 = 1;
;  2846          3001  4
;  2847          3002  4         R0 = .VALUE [0];
;  2848          3003  4         R1 = .VALUE [1];
;  2849          3004  4         RETURN;
;  2850          3005  3         END;
;  2851          3006  2         END;
;  2852          3007  2
;  2853          3008  2     IF (.DESCRIP [DSC$B_CLASS] NEQU DSC$K_CLASS_A) THEN BAS$$STOP (BAS$K_NOTIMP);
;  2854          3009  2
;  2855          3010  2     !+
;  2856          3011  2     ! Return the array element as our value.
;  2857          3012  2     !-
;  2858          3013  3     BEGIN
;  2859          3014  3
;  2860          3015  3     REGISTER
;  2861          3016  3         R0 = 0,
;  2862          3017  3         R1 = 1;
;  2863          3018  3
;  2864          3019  3     BAS$$COPY_G_R1 (.VALUE_LOCATION, VALUE);
;  2865          3020  3     R0 = .VALUE [0];
```

```
;  2866          3021  3     R1 = .VALUE [1];
;  2867          3022  3     RETURN;
;  2868          3023  2     END;
;  2869          3024  1     END;                                    ! end of BAS$FET_FA_G_R8


                                                 5E              14  C2 00000 BAS$FET_FA_G_R8::
                                                                             SUBL2   #20, SP                                 ; 2850
                                                                 51  DD 00003   PUSHL   R1
                                                                 54  D0 00005   MOVL    R0, R4
                                                                 53      0B  A4  9A 00008   MOVZBL  11(DESCRIP), R3          ; 2904
                                                                             05  13 0000C   BEQL    1$
                                                                 02          53  91 0000E   CMPB    R3, #2
                                                                             0B  1B 00011   BLEQU   2$
                                                                 7E      00G 8F  9A 00013 1$:  MOVZBL  #BAS$K_ONEOR_TWO, -(SP)
                                                     00000000G   00          01  FB 00017   CALLS   #1, BA$$$STOP
                                                                 18      02  A4  91 0001E 2$:  CMPB    2(DESCRIP), #27     ; 2910
                                                                             1B  13 00022   BEQL    4$
                                                                 18      02  A4  91 00024   CMPB    2(DESCRIP), #24     ; 2912
                                                                             0A  12 00028   BNEQ    3$
                                                                 56      04  A4  D0 0002A   MOVL    4(DESCRIP), TEMP_DESCRIP ; 2919
                                                                 1B      02  A6  91 0002E   CMPB    2(TEMP_DESCRIP), #27 ; 2920
                                                                             0B  13 00032   BEQL    4$
                                                                 7E      00G 8F  9A 00034 3$:  MOVZBL  #BAS$K_ARGDONMAT, -(SP) ; 2926
                                                     00000000G   00          01  FB 00038   CALLS   #1, BA$$$STOP
                                                 05      0A  A4              06  E1 0003F 4$:  BBC     #6, 10(DESCRIP), 5$ ; 2932
                                                                 0A  A4      95  00044   TSTB    10(DESCRIP)             ; 2933
                                                                             0B  19 00047   BLSS    6$
                                                                 7E      00G 8F  9A 00049 5$:  MOVZBL  #BAS$K_ARGDONMAT, -(SP) ; 2935
                                                     00000000G   00          01  FB 0004D   CALLS   #1, BA$$$STOP
                                                         08      AE  14  A4  9E 00054 6$:  MOVAB   20(R4), MULTIPLIERS     ; 2937
                                                                 55  14 A443  DE 00059   MOVAL   20(DESCRIP)[R3], BOUNDS  ; 2938
                                                 0C      0A  A4              05  E1 0005E   BBC     #5, 10(DESCRIP), 7$ ; 2944
                                                                 51          53  D0 00063   MOVL    R3, LOW_INDEX       ; 2947
                                                                 50          01  D0 00066   MOVL    #1, HIGH_INDEX      ; 2948
                                                         04      AE          01  CE 00069   MNEGL   #1, INDEX_INCR      ; 2949
                                                                             0A  11 0006D   BRB     8$                  ; 2944
                                                                 51          01  D0 0006F 7$:  MOVL    #1, LOW_INDEX       ; 2953
                                                                 50          53  D0 00072   MOVL    R3, HIGH_INDEX      ; 2954
                                                         04      AE          01  D0 00075   MOVL    #1, INDEX_INCR      ; 2955
                                                 53              51  04      AE  C3 00079 8$:  SUBL3   INDEX_INCR, LOW_INDEX, INDEX_NUMBER ; 2958
                                                                             57  D4 0007E   CLRL    VALUE_LOCATION      ; 2962
                                                         0C      AE  04 BE40 9E 00080   MOVAB   @INDEX_INCR[HIGH_INDEX], 12(SP) ; 2964
                                                                 53  04      AE  C0 00086 9$:  ADDL2   INDEX_INCR, INDEX_NUMBER
                                                         0C      AE          53  D1 0008A   CMPL    INDEX_NUMBER, 12(SP)
                                                                             3A  13 0008E   BEQL    14$
                                                                 01          53  D1 00090   CMPL    INDEX_NUMBER, #1    ; 2966
                                                                             05  12 00093   BNEQ    10$
                                                                 58          6E  D0 00095   MOVL    INDEX1, INDEX_VALUE
                                                                             03  11 00098   BRB     11$
                                                                 58          52  D0 0009A 10$:  MOVL    INDEX2, INDEX_VALUE
                                                 50              53          01  78 0009D 11$:  ASHL    #1, INDEX_NUMBER, R0 ; 2968
                                                             F8  A540        58  D1 000A1   CMPL    INDEX_VALUE, -8(BOUNDS)[R0]
                                                                             07  19 000A6   BLSS    12$
```

```
                FC A540              58  D1 000AB              CMPL    INDEX_VALUE, -4(BOUNDS)[R0]                    ; 2969
                                     0B  15 000AD              BLEQ    13$
                      7E       00G   8F  9A 000AF  12$:       MOVZBL  #BAS$K_SUBOUTRAN, -(SP)                        ; 2971
          00000000G   00             01  FB 000B3              CALLS   #1, BAS$$STOP
          51    08    AE             04  C3 000BA  13$:       SUBL3   #4, MULTIPLIERS, R1                            ; 2973
          50          57         6143  C5 000BF              MULL3   (R1)[INDEX_NUMBER], VALUE_LOCATION, R0
          57          50             58  C1 000C4              ADDL3   INDEX_VALUE, R0, VALUE_LOCATION
                                     8C  11 000C8              BRB     9$                                             ; 2964
                      50             64  3C 000CA  14$:       MOVZWL  (DESCRIP), R0                                  ; 2976
                      50             57  C4 000CD              MULL2   VALUE_LOCATION, R0
          57          50         10  A4  C1 000D0              ADDL3   16(DESCRIP), R0, VALUE_LOCATION
                      18         02  A4  91 000D5              CMPB    2(DESCRIP), #24                                ; 2977
                                     07  12 000D9              BNEQ    15$
                      56             57  D0 000DB              MOVL    VALUE_LOCATION, TEMP_DESCRIP                   ; 2984
                      57         04  A6  D0 000DE              MOVL    4(TEMP_DESCRIP), VALUE_LOCATION                ; 2985
                BF    8F         03  A4  91 000E2  15$:       CMPB    3(DESCRIP), #191                               ; 2992
                                     10  12 000E7              BNEQ    16$
                                 10  AE  9F 000E9              PUSHAB  VALUE                                          ; 2995
                               0090  8F  BB 000EC              PUSHR   #^M<R4,R7>
          00000000G   00             03  FB 000F0              CALLS   #3, BAS$$VA_FETCH
                                     1E  11 000F7              BRB     18$                                            ; 3002
                      04         03  A4  91 000F9  16$:       CMPB    3(DESCRIP), #4                                 ; 3008
                                     0B  13 000FD              BEQL    17$
                      7E       00G   8F  9A 000FF              MOVZBL  #BAS$K_NOTIMP, -(SP)
          00000000G   00             01  FB 00103              CALLS   #1, BAS$$STOP
                      51         10  AE  9E 0010A  17$:       MOVAB   VALUE, R1                                      ; 3019
                      50             57  D0 0010E              MOVL    VALUE_LOCATION, R0
                           00000000G   00  16 00111              JSB     BAS$$COPY_G_R1
                      50         10  AE  7D 00117  18$:       MOVQ    VALUE, R0                                      ; 3020
                      5E             18  C0 0011B              ADDL2   #24, SP                                        ; 3024
                                     05  0011E                RSB                                                    ;

; Routine Size:  287 bytes,    Routine Base:  _BAS$CODE + 1343

; 2870          3025  1
```

```
;  2872          3026  1 GLOBAL ROUTINE BASSFET_FA_H_R8 (                      ! Fetch an h floating
;  2873          3027  1         DESCRIP : REF BLOCK[8, BYTE],                ! The descriptor to fetch from
;  2874          3028  1         INDEX1,                                      ! First index
;  2875          3029  1         INDEX2                                       ! Second index
;  2876          3030  1     ) : VA_JSB NOVALUE =
;  2877          3031  1
;  2878          3032  1 !++
;  2879          3033  1 ! FUNCTIONAL DESCRIPTION:
;  2880          3034  1 !
;  2881          3035  1 !       Fetch an h floating number from an array or virtual array.
;  2882          3036  1 !
;  2883          3037  1 ! FORMAL PARAMETERS:
;  2884          3038  1 !
;  2885          3039  1 !       DESCRIP.rd.da   The descriptor of the array or virtual array
;  2886          3040  1 !       INDEX1.rl.v     The first index into the array
;  2887          3041  1 !       INDEX2.rl.v     The second index into the array
;  2888          3042  1 !
;  2889          3043  1 ! IMPLICIT INPUTS:
;  2890          3044  1 !
;  2891          3045  1 !       NONE
;  2892          3046  1 !
;  2893          3047  1 ! IMPLICIT OUTPUTS:
;  2894          3048  1 !
;  2895          3049  1 !       NONE
;  2896          3050  1 !
;  2897          3051  1 ! ROUTINE VALUE:
;  2898          3052  1 ! COMPLETION CODES:
;  2899          3053  1 !
;  2900          3054  1 !       The h floating number from the array or virtual array
;  2901          3055  1 !
;  2902          3056  1 ! SIDE EFFECTS:
;  2903          3057  1 !
;  2904          3058  1 !       Signals if an error is encountered.
;  2905          3059  1 !
;  2906          3060  1 !--
;  2907          3061  1
;  2908          3062  2     BEGIN
;  2909          3063  2
;  2910          3064  2     LOCAL
;  2911          3065  2         BOUNDS : REF VECTOR,
;  2912          3066  2         MULTIPLIERS : REF VECTOR,
;  2913          3067  2         LOW_INDEX,
;  2914          3068  2         HIGH_INDEX,
;  2915          3069  2         INDEX_INCR,
;  2916          3070  2         VALUE_LOCATION,
;  2917          3071  2         INDEX_VALUE,
;  2918          3072  2         INDEX_NUMBER,
;  2919          3073  2         VALUE : VECTOR [4],
;  2920          3074  2         TEMP_DESCRIP: REF BLOCK[,BYTE];
;  2921          3075  2
;  2922          3076  2 !+
;  2923          3077  2 ! Be sure the array has at least one but no more than two dimensions.
;  2924          3078  2 !-
;  2925          3079  2
;  2926          3080  2     IF ((.DESCRIP [DSC$B_DIMCT] LSSU 1) OR (.DESCRIP [DSC$B_DIMCT] GTRU 2)) THEN BAS$$STOP (BAS$K_ONEOR_TWO)
;  2927          3081  2
;  2928          3082  2 !+
```

```
: 2929          3083  2 ! Be sure this array or virtual array holds h floating numbers.
: 2930          3084  2 !-
: 2931          3085  2
: 2932          3086  2     IF (.DESCRIP [DSC$B_DTYPE] NEQU DSC$K_DTYPE_H)
: 2933          3087  2     THEN
: 2934          3088  2         IF (.DESCRIP [DSC$B_DTYPE] EQLU DSC$K_DTYPE_DSC)
: 2935          3089  2         THEN
: 2936          3090  2             !+
: 2937          3091  2             ! Special handling for dynamically mapped arrays.
: 2938          3092  2             !-
: 2939          3093  2             BEGIN
: 2940          3094  2
: 2941          3095  2             TEMP_DESCRIP = .DESCRIP [DSC$A_POINTER];
: 2942          3096  2             IF (.TEMP_DESCRIP [DSC$B_DTYPE] NEQU DSC$K_DTYPE_H)
: 2943          3097  2             THEN
: 2944          3098  2                 BAS$$STOP (BAS$K_ARGDONMAT);
: 2945          3099  2
: 2946          3100  2             END
: 2947          3101  2         ELSE
: 2948          3102  2             BAS$$STOP (BAS$K_ARGDONMAT);
: 2949          3103  2
: 2950          3104  2     !+
: 2951          3105  2     ! The coefficients and bounds must be present
: 2952          3106  2     !-
: 2953          3107  2
: 2954          3108  2     IF ( NOT (.DESCRIP [DSC$V_FL_COEFF] AND     !
: 2955          3109  2         .DESCRIP [DSC$V_FL_BOUNDS]))
: 2956          3110  2     THEN
: 2957          3111  2         BAS$$STOP (BAS$K_ARGDONMAT);
: 2958          3112  2
: 2959          3113  2     MULTIPLIERS = DESCRIP [DSC$L_M1];
: 2960          3114  2     BOUNDS = DESCRIP [DSC$L_M1] + (%UPVAL*.DESCRIP [DSC$B_DIMCT]);
: 2961          3115  2     !+
: 2962          3116  2     ! Compute the lower and upper index numbers based on how the array
: 2963          3117  2     ! is stored.
: 2964          3118  2     !-
: 2965          3119  2
: 2966          3120  2     IF (.DESCRIP [DSC$V_FL_COLUMN])
: 2967          3121  2     THEN
: 2968          3122  2         BEGIN
: 2969          3123  2         LOW_INDEX = .DESCRIP [DSC$B_DIMCT];
: 2970          3124  2         HIGH_INDEX = 1;
: 2971          3125  2         INDEX_INCR = -1;
: 2972          3126  2         END
: 2973          3127  2     ELSE
: 2974          3128  2         BEGIN
: 2975          3129  2         LOW_INDEX = 1;
: 2976          3130  2         HIGH_INDEX = .DESCRIP [DSC$B_DIMCT];
: 2977          3131  2         INDEX_INCR = 1;
: 2978          3132  2         END;
: 2979          3133  2
: 2980          3134  2     INDEX_NUMBER = .LOW_INDEX - .INDEX_INCR;
: 2981          3135  2     !+
: 2982          3136  2     ! Compute the linear index from the indices provided.
: 2983          3137  2     !-
: 2984          3138  2     VALUE_LOCATION = 0;
: 2985          3139  2
```

```
;  2986          3140  2      WHILE ((INDEX_NUMBER = .INDEX_NUMBER + .INDEX_INCR) NEQ (.HIGH_INDEX + .INDEX_INCR)) DO
;  2987          3141  2          BEGIN
;  2988          3142  2          INDEX_VALUE = (IF (.INDEX_NUMBER EQL 1) THEN .INDEX1 ELSE .INDEX2);
;  2989          3143  2
;  2990          3144  2          IF ((.INDEX_VALUE LSS .BOUNDS [(.INDEX_NUMBER - 1)*2])  !
;  2991          3145  2              OR (.INDEX_VALUE GTR .BOUNDS [((.INDEX_NUMBER - 1)*2) + 1]))
;  2992          3146  2          THEN
;  2993          3147  2              BAS$$STOP (BAS$K_SUBOUTRAN);
;  2994          3148  2
;  2995          3149  2          VALUE_LOCATION = (.VALUE_LOCATION+.MULTIPLIERS [.INDEX_NUMBER - 1]) + .INDEX_VALUE;
;  2996          3150  2          END;
;  2997          3151  2
;  2998          3152  2      VALUE_LOCATION = (.VALUE_LOCATION+.DESCRIP [DSC$W_LENGTH]) + .DESCRIP [DSC$A_A0];
;  2999          3153  2      IF .DESCRIP [DSC$B_DTYPE] EQLU DSC$K_DTYPE_DSC
;  3000          3154  2      THEN
;  3001          3155  2          !+
;  3002          3156  2          ! Special handling for dynamically mapped arrays.
;  3003          3157  2          !-
;  3004          3158  2          BEGIN
;  3005          3159  2
;  3006          3160  2          TEMP_DESCRIP = .VALUE_LOCATION;
;  3007          3161  2          VALUE_LOCATION = .TEMP_DESCRIP [DSC$A_POINTER];
;  3008          3162  2
;  3009          3163  2          END;
;  3010          3164  2  !+
;  3011          3165  2  ! Special handling for virtual arrays.
;  3012          3166  2  !-
;  3013          3167  2
;  3014          3168  3      IF (.DESCRIP [DSC$B_CLASS] EQLU DSC$K_CLASS_BFA)
;  3015          3169  2      THEN
;  3016          3170  3          BEGIN
;  3017          3171  3          BAS$$VA_FETCH (.DESCRIP, .VALUE_LOCATION, VALUE);
;  3018          3172  4          BEGIN
;  3019          3173  4
;  3020          3174  4          REGISTER
;  3021          3175  4              R0 = 0,
;  3022          3176  4              R1 = 1,
;  3023          3177  4              R2 = 2,
;  3024          3178  4              R3 = 3;
;  3025          3179  4
;  3026          3180  4          R0 = .VALUE [0];
;  3027          3181  4          R1 = .VALUE [1];
;  3028          3182  4          R2 = .VALUE [2];
;  3029          3183  4          R3 = .VALUE [3];
;  3030          3184  4          RETURN;
;  3031          3185  3          END;
;  3032          3186  2          END;
;  3033          3187  2
;  3034          3188  2      IF (.DESCRIP [DSC$B_CLASS] NEQU DSC$K_CLASS_A) THEN BAS$$STOP (BAS$K_NOTIMP);
;  3035          3189  2
;  3036          3190  2  !+
;  3037          3191  2  ! Return the array element as our value.
;  3038          3192  2  !-
;  3039          3193  3      BEGIN
;  3040          3194  3
;  3041          3195  3      REGISTER
;  3042          3196  3          R0 = 0,
```

```
;  3043          3197  3          R1 = 1,
;  3044          3198  3          R2 = 2,
;  3045          3199  3          R3 = 3;
;  3046          3200  3
;  3047          3201  3      BAS$$COPY_H_R3 (.VALUE_LOCATION, VALUE);
;  3048          3202  3      R0 = .VALUE [0];
;  3049          3203  3      R1 = .VALUE [1];
;  3050          3204  3      R2 = .VALUE [2];
;  3051          3205  3      R3 = .VALUE [3];
;  3052          3206  3      RETURN;
;  3053          3207  2      END;
;  3054          3208  1      END;                                    ! end of BAS$FET_FA_H_R8


                                                    5E            1C  C2 00000 BAS$FET_FA_H_R8::
                                                                              SUBL2   #28, SP                                 ; 3026
                                                                  51  DD 00003          PUSHL   R1
                                                                  54  D0 00005          MOVL    R0, R4
                                                                  53      0B  A4  9A 00008          MOVZBL  11(DESCRIP), R3                 ; 3080
                                                                              05  13 0000C          BEQL    1$
                                                                  02              53  91 0000E          CMPB    R3, #2
                                                                              0B  1B 00011          BLEQU   2$
                                                                  7E      00G 8F  9A 00013 1$:      MOVZBL  #BAS$K_ONEOR_TWO, -(SP)
                                                      00000000G   00          01  FB 00017          CALLS   #1, BAS$$STOP
                                                                  1C      02  A4  91 0001E 2$:      CMPB    2(DESCRIP), #28         ; 3086
                                                                              1B  13 00022          BEQL    4$
                                                                  18      02  A4  91 00024          CMPB    2(DESCRIP), #24         ; 3088
                                                                              0A  12 00028          BNEQ    3$
                                                                  56      04  A4  D0 0002A          MOVL    4(DESCRIP), TEMP_DESCRIP ; 3095
                                                                  1C      02  A6  91 0002E          CMPB    2(TEMP_DESCRIP), #28    ; 3096
                                                                              0B  13 00032          BEQL    4$
                                                                  7E      00G 8F  9A 00034 3$:      MOVZBL  #BAS$K_ARGDONMAT, -(SP) ; 3102
                                                      00000000G   00          01  FB 00038          CALLS   #1, BAS$$STOP
                                                05        0A    A4              06  E1 0003F 4$:      BBC     #6, 10(DESCRIP), 5$     ; 3108
                                                                  0A      A4      95 00044          TSTB    10(DESCRIP)             ; 3109
                                                                              08  19 00047          BLSS    6$
                                                                  7E      00G 8F  9A 00049 5$:      MOVZBL  #BAS$K_ARGDONMAT, -(SP) ; 3111
                                                      00000000G   00          01  FB 0004D          CALLS   #1, BAS$$STOP
                                                            08    AE    14  A4  9E 00054 6$:      MOVAB   20(R4), MULTIPLIERS     ; 3113
                                                                  55    14  A443 DE 00059          MOVAL   20(DESCRIP)[R3], BOUNDS ; 3114
                                                0C        0A    A4              05  E1 0005E          BBC     #5, 10(DESCRIP), 7$     ; 3120
                                                                  51              53  D0 00063          MOVL    R3, LOW_INDEX           ; 3123
                                                                  50              01  D0 00066          MOVL    #1, HIGH_INDEX          ; 3124
                                                            04    AE              01  CE 00069          MNEGL   #1, INDEX_INCR          ; 3125
                                                                              0A  11 0006D          BRB     8$                      ; 3120
                                                                  51              01  D0 0006F 7$:      MOVL    #1, LOW_INDEX           ; 3129
                                                                  50              53  D0 00072          MOVL    R3, HIGH_INDEX          ; 3130
                                                            04    AE              01  D0 00075          MOVL    #1, INDEX_INCR          ; 3131
                                                53              51      04  AE  C3 00079 8$:      SUBL3   INDEX_INCR, LOW_INDEX, INDEX_NUMBER ; 3134
                                                                              57  D4 0007E          CLRL    VALUE_LOCATION          ; 3138
                                                            0C    AE      04 BE40 9E 00080          MOVAB   @INDEX_INCR[HIGH_INDEX], 12(SP) ; 3140
                                                                  53      04  AE  C0 00086 9$:      ADDL2   INDEX_INCR, INDEX_NUMBER
                                                            0C    AE              53  D1 0008A          CMPL    INDEX_NUMBER, 12(SP)
                                                                              3A  13 0008E          BEQL    14$
```

```
                    01          53  D1 00090          CMPL    INDEX_NUMBER, #1                                    ;3142
                                05  12 00093          BNEQ    10$
                    58          6E  D0 00095          MOVL    INDEX1, INDEX_VALUE
                                03  11 00098          BRB     11$
                    58          52  D0 0009A 10$:      MOVL    INDEX2, INDEX_VALUE
50                  53          01  78 0009D 11$:      ASHL    #1, INDEX_NUMBER, R0                                ;3144
                 F8 A540        58  D1 000A1          CMPL    INDEX_VALUE, -8(BOUNDS)[R0]
                                07  19 000A6          BLSS    12$
                 FC A540        58  D1 000A8          CMPL    INDEX_VALUE, -4(BOUNDS)[R0]                         ;3145
                                08  15 000AD          BLEQ    13$
                    7E      00G 8F  9A 000AF 12$:      MOVZBL  #BAS$K_SUBOUTRAN, -(SP)                             ;3147
          00000000G 00          01  FB 000B3          CALLS   #1, BAS$$STOP
51          08      AE          04  C3 000BA 13$:      SUBL3   #4, MULTIPLIERS, R1                                 ;3149
50                  57      6143  C5 000BF          MULL3   (R1)[INDEX_NUMBER], VALUE_LOCATION, R0
57                  50          58  C1 000C4          ADDL3   INDEX_VALUE, R0, VALUE_LOCATION
                                BC  11 000C8          BRB     9$                                                  ;3140
                    50          64  3C 000CA 14$:      MOVZWL  (DESCRIP), R0                                       ;3152
                    50          57  C4 000CD          MULL2   VALUE_LOCATION, R0
57                  50      10  A4  C1 000D0          ADDL3   16(DESCRIP), R0, VALUE_LOCATION
                    18      02  A4  91 000D5          CMPB    2(DESCRIP), #24                                     ;3153
                                07  12 000D9          BNEQ    15$
                    56          57  D0 000DB          MOVL    VALUE_LOCATION, TEMP_DESCRIP                        ;3160
                    57      04  A6  D0 000DE          MOVL    4(TEMP_DESCRIP), VALUE_LOCATION                     ;3161
                 BF     8F      03  A4  91 000E2 15$:      CMPB    3(DESCRIP), #191                                ;3168
                                10  12 000E7          BNEQ    16$
                            10  AE  9F 000E9          PUSHAB  VALUE                                               ;3171
                          0090  8F  BB 000EC          PUSHR   #^M<R4,R7>
          00000000G 00          03  FB 000F0          CALLS   #3, BAS$$VA_FETCH
                                1E  11 000F7          BRB     18$                                                 ;3180
                    04      03  A4  91 000F9 16$:      CMPB    3(DESCRIP), #4                                      ;3188
                                0B  13 000FD          BEQL    17$
                    7E      00G 8F  9A 000FF          MOVZBL  #BAS$K_NOTIMP, -(SP)
          00000000G 00          01  FB 00103          CALLS   #1, BAS$$STOP
                    51      10  AE  9E 0010A 17$:      MOVAB   VALUE, R1                                           ;3201
                    50          57  D0 0010E          MOVL    VALUE_LOCATION, R0
                 00000000G  00  16  00111          JSB     BAS$$COPY_H_R3
                    50      10  AE  7D 00117 18$:      MOVQ    VALUE, R0                                           ;3202
                    52      18  AE  7D 0011B          MOVQ    VALUE+8, R2                                         ;3204
                    5E          20  C0 0011F          ADDL2   #32, SP                                             ;3208
                                05  00122          RSB

; Routine Size:  291 bytes,    Routine Base:  _BAS$CODE + 1462

; 3055          3209  1
```

```
;  3057          3210  1 GLOBAL ROUTINE BAS$STO_FA_B_R8 (                      ! Store a byte
;  3058          3211  1         VALUE,                                        ! The value to store
;  3059          3212  1         DESCRIP : REF BLOCK [8, BYTE],                ! The descriptor to store into
;  3060          3213  1         INDEX1,                                       ! First index
;  3061          3214  1         INDEX2                                        ! Second index
;  3062          3215  1     ) : VA_JSB NOVALUE =
;  3063          3216  1
;  3064          3217  1 !++
;  3065          3218  1 ! FUNCTIONAL DESCRIPTION:
;  3066          3219  1 !
;  3067          3220  1 !       Store a byte in an array or virtual array.
;  3068          3221  1 !
;  3069          3222  1 ! FORMAL PARAMETERS:
;  3070          3223  1 !
;  3071          3224  1 !       VALUE.rb.v      The value to store
;  3072          3225  1 !       DESCRIP.rw.da   The descriptor of the array or virtual array
;  3073          3226  1 !       INDEX1.rl.v     The first index into the array
;  3074          3227  1 !       INDEX2.rl.v     The second index into the array
;  3075          3228  1 !
;  3076          3229  1 ! IMPLICIT INPUTS:
;  3077          3230  1 !
;  3078          3231  1 !       NONE
;  3079          3232  1 !
;  3080          3233  1 ! IMPLICIT OUTPUTS:
;  3081          3234  1 !
;  3082          3235  1 !       NONE
;  3083          3236  1 !
;  3084          3237  1 ! ROUTINE VALUE:
;  3085          3238  1 ! COMPLETION CODES:
;  3086          3239  1 !
;  3087          3240  1 !       NONE
;  3088          3241  1 !
;  3089          3242  1 ! SIDE EFFECTS:
;  3090          3243  1 !
;  3091          3244  1 !       Signals if an error is encountered.
;  3092          3245  1 !
;  3093          3246  1 !--
;  3094          3247  1
;  3095          3248  2     BEGIN
;  3096          3249  2
;  3097          3250  2     LOCAL
;  3098          3251  2         BOUNDS : REF VECTOR,
;  3099          3252  2         MULTIPLIERS : REF VECTOR,
;  3100          3253  2         LOW_INDEX,
;  3101          3254  2         HIGH_INDEX,
;  3102          3255  2         INDEX_INCR,
;  3103          3256  2         VALUE_LOCATION,
;  3104          3257  2         INDEX_VALUE,
;  3105          3258  2         INDEX_NUMBER;
;  3106          3259  2
;  3107          3260  2 !+
;  3108          3261  2 ! Be sure the array has at least one but no more than two dimensions.
;  3109          3262  2 !-
;  3110          3263  2
;  3111          3264  2     IF ((.DESCRIP [DSC$B_DIMCT] LSSU 1) OR (.DESCRIP [DSC$B_DIMCT] GTRU 2)) THEN BAS$$STOP (BAS$K_ONEOR_TWO)
;  3112          3265  2
;  3113          3266  2 !+
```

```
;  3114          3267  2 ! Be sure this array or virtual array holds words.
;  3115          3268  2 !-
;  3116          3269  2
;  3117          3270  2     IF (.DESCRIP [DSC$B_DTYPE] NEQU DSC$K_DTYPE_B) THEN BAS$$STOP (BAS$K_ARGDONMAT);
;  3118          3271  2
;  3119          3272  2 !+
;  3120          3273  2 ! The coefficients and bounds must be present
;  3121          3274  2 !-
;  3122          3275  2
;  3123          3276  2     IF ( NOT (.DESCRIP [DSC$V_FL_COEFF] AND      !
;  3124          3277  2          .DESCRIP [DSC$V_FL_BOUNDS]))
;  3125          3278  2     THEN
;  3126          3279  2         BAS$$STOP (BAS$K_ARGDONMAT);
;  3127          3280  2
;  3128          3281  2     MULTIPLIERS = DESCRIP [DSC$L_M1];
;  3129          3282  2     BOUNDS = DESCRIP [DSC$L_M1] + (%UPVAL*.DESCRIP [DSC$B_DIMCT]);
;  3130          3283  2 !+
;  3131          3284  2 ! Compute the lower and upper index numbers based on how the array
;  3132          3285  2 ! is stored.
;  3133          3286  2 !-
;  3134          3287  2
;  3135          3288  2     IF (.DESCRIP [DSC$V_FL_COLUMN])
;  3136          3289  2     THEN
;  3137          3290  2         BEGIN
;  3138          3291  2         LOW_INDEX = .DESCRIP [DSC$B_DIMCT];
;  3139          3292  2         HIGH_INDEX = 1;
;  3140          3293  2         INDEX_INCR = -1;
;  3141          3294  2         END
;  3142          3295  2     ELSE
;  3143          3296  2         BEGIN
;  3144          3297  2         LOW_INDEX = 1;
;  3145          3298  2         HIGH_INDEX = .DESCRIP [DSC$B_DIMCT];
;  3146          3299  2         INDEX_INCR = 1;
;  3147          3300  2         END;
;  3148          3301  2
;  3149          3302  2     INDEX_NUMBER = .LOW_INDEX - .INDEX_INCR;
;  3150          3303  2 !+
;  3151          3304  2 ! Compute the linear index from the indices provided.
;  3152          3305  2 !-
;  3153          3306  2     VALUE_LOCATION = 0;
;  3154          3307  2
;  3155          3308  2     WHILE ((INDEX_NUMBER = .INDEX_NUMBER + .INDEX_INCR) NEQ (.HIGH_INDEX + .INDEX_INCR)) DO
;  3156          3309  2         BEGIN
;  3157          3310  2         INDEX_VALUE = (IF (.INDEX_NUMBER EQL 1) THEN .INDEX1 ELSE .INDEX2);
;  3158          3311  2
;  3159          3312  2         IF ((.INDEX_VALUE LSS .BOUNDS [(.INDEX_NUMBER - 1)*2])  !
;  3160          3313  2             OR (.INDEX_VALUE GTR .BOUNDS [((.INDEX_NUMBER - 1)*2) + 1]))
;  3161          3314  2         THEN
;  3162          3315  2             BAS$$STOP (BAS$K_SUBOUTRAN);
;  3163          3316  2
;  3164          3317  2         VALUE_LOCATION = (.VALUE_LOCATION*.MULTIPLIERS [.INDEX_NUMBER - 1]) + .INDEX_VALUE;
;  3165          3318  2         END;
;  3166          3319  2
;  3167          3320  2     VALUE_LOCATION = (.VALUE_LOCATION*.DESCRIP [DSC$W_LENGTH]) + .DESCRIP [DSC$A_A0];
;  3168          3321  2 !+
;  3169          3322  2 ! Special handling for virtual arrays.
;  3170          3323  2 !-
```

```
;  3171          3324  2
;  3172          3325  2      IF (.DESCRIP [DSC$B_CLASS] EQLU DSC$K_CLASS_BFA)
;  3173          3326  2      THEN
;  3174          3327  2          BEGIN
;  3175          3328  2          BAS$$VA_STORE (.DESCRIP, .VALUE_LOCATION, VALUE);
;  3176          3329  2          END
;  3177          3330  2      ELSE
;  3178          3331  2          BEGIN
;  3179          3332  2
;  3180          3333  2          IF (.DESCRIP [DSC$B_CLASS] NEQU DSC$K_CLASS_A) THEN BAS$$STOP (BAS$K_NOTIMP);
;  3181          3334  2
;  3182          3335  2      !+
;  3183          3336  2      ! Store the value provided into the array
;  3184          3337  2      !-
;  3185          3338  2          BLOCK [.VALUE_LOCATION, 0, 0, %BPUNIT, 1] = .VALUE;
;  3186          3339  2          END;
;  3187          3340  2
;  5188          3341  1      END;                                    ! end of BAS$STO_FA_B_R8


                                5E              10  C2 00000 BAS$STO_FA_B_R8::
                                                              SUBL2   #16, SP                                 ;  3210
                                56              51  D0 00003         MOVL    R1, R6                          ;
                             0C AE              50  D0 00006         MOVL    R0, VALUE                       ;
                                54        0B    A6  9A 0000A         MOVZBL  11(DESCRIP), R4                 ;  3264
                                                  05  13 0000E         BEQL    1$                              ;
                                02              54  91 00010         CMPB    R4, #2                          ;
                                                  0B  1B 00013         BLEQU   2$                              ;
                                7E        00G   8F  9A 00015 1$:      MOVZBL  #BAS$K_ONEOR_TWO, -(SP)         ;
                      00000000G 00              01  FB 00019         CALLS   #1, BAS$$STOP                   ;
                                06        02    A6  91 00020 2$:      CMPB    2(DESCRIP), #6                  ;  3270
                                                  0B  13 00024         BEQL    3$                              ;
                                7E        00G   8F  9A 00026         MOVZBL  #BAS$K_ARGDONMAT, -(SP)         ;
                      00000000G 00              01  FB 0002A         CALLS   #1, BAS$$STOP                   ;
                          05        0A    A6                  06  E1 00031 3$:      BBC     #6, 10(DESCRIP), 4$     ;  3276
                                          0A    A6  95 00036         TSTB    10(DESCRIP)                     ;  3277
                                                  0B  19 00039         BLSS    5$                              ;
                                7E        00G   8F  9A 0003B 4$:      MOVZBL  #BAS$K_ARGDONMAT, -(SP)         ;  3279
                      00000000G 00              01  FB 0003F         CALLS   #1, BAS$$STOP                   ;
                                   04     AE    14  A6  9E 00046 5$:      MOVAB   20(R6), MULTIPLIERS             ;  3281
                                57              14 A644 DE 0004B         MOVAL   20(DESCRIP)[R4], BOUNDS         ;  3282
                          08        0A    A6                  05  E1 00050         BBC     #5, 10(DESCRIP), 6$     ;  3288
                                51                    54  D0 00055         MOVL    R4, LOW_INDEX           ;  3291
                                50                    01  D0 00058         MOVL    #1, HIGH_INDEX          ;  3292
                                6E                    01  CE 0005B         MNEGL   #1, INDEX_INCR          ;  3293
                                                      09  11 0005E         BRB     7$                      ;  3288
                                51                    01  D0 00060 6$:      MOVL    #1, LOW_INDEX           ;  3297
                                50                    54  D0 00063         MOVL    R4, HIGH_INDEX          ;  3298
                                6E                    01  D0 00066         MOVL    #1, INDEX_INCR          ;  3299
                             55                 51    6E  C3 00069 7$:      SUBL3   INDEX_INCR, LOW_INDEX, INDEX_NUMBER     ;  3302
                                                      54  D4 0006D         CLRL    VALUE_LOCATION          ;  3306
                          08 AE                 50    6E  C1 0006F         ADDL3   INDEX_INCR, HIGH_INDEX, 8(SP)   ;  3308
                                                55    6E  C0 00074 8$:      ADDL2   INDEX_INCR, INDEX_NUMBER        ;
                                   08     AE    55  D1 00077         CMPL    INDEX_NUMBER, 8(SP)             ;
```

```
                                    3A  13 0007B              BEQL    13$                                                     ;
                          01        55  D1 0007D              CMPL    INDEX_NUMBER, #1                                        ; 3310
                                    05  12 00080              BNEQ    9$                                                      ;
                          58        52  D0 00082              MOVL    INDEX1, INDEX_VALUE                                     ;
                                    03  11 00085              BRB     10$                                                     ;
                          58        53  D0 00087  9$:         MOVL    INDEX2, INDEX_VALUE                                     ;
                    50    55        01  78 0008A  10$:        ASHL    #1, INDEX_NUMBER, R0                                    ; 3312
                FB A740             58  D1 0008E              CMPL    INDEX_VALUE, -8(BOUNDS)[R0]                             ;
                                    07  19 00093              BLSS    11$                                                     ;
                FC A740             58  D1 00095              CMPL    INDEX_VALUE, -4(BOUNDS)[R0]                             ; 3313
                                    0B  15 0009A              BLEQ    12$                                                     ;
                          7E    00G 8F  9A 0009C  11$:        MOVZBL  #BAS$K_SUBOUTRAN, -(SP)                                 ; 3315
          00000000G       00        01  FB 000A0              CALLS   #1, BAS$$STOP                                           ;
                    51    04  AE        04  C3 000A7  12$:    SUBL3   #4, MULTIPLIERS, R1                                     ; 3317
                    50        54    6145  C5 000AC              MULL3   (R1)[INDEX_NUMBER], VALUE_LOCATION, R0                  ;
                    54        50        58  C1 000B1              ADDL3   INDEX_VALUE, R0, VALUE_LOCATION                         ;
                                        BD  11 000B5              BRB     8$                                                      ; 3308
                              50        66  3C 000B7  13$:        MOVZWL  (DESCRIP), R0                                           ; 3320
                              50        54  C4 000BA              MULL2   VALUE_LOCATION, R0                                      ;
                    54        50  10    A6  C1 000BD              ADDL3   16(DESCRIP), R0, VALUE_LOCATION                         ;
                FB    8F        03    A6  91 000C2              CMPB    3(DESCRIP), #191                                        ; 3325
                                        10  12 000C7              BNEQ    14$                                                     ;
                                    0C  AE  9F 000C9              PUSHAB  VALUE                                                   ; 3328
                                        54  DD 000CC              PUSHL   VALUE_LOCATION                                          ;
                                        56  DD 000CE              PUSHL   DESCRIP                                                 ;
          00000000G       00            03  FB 000D0              CALLS   #3, BAS$$VA_STORE                                       ;
                                        15  11 000D7              BRB     16$                                                     ; 3325
                              04    03  A6  91 000D9  14$:        CMPB    3(DESCRIP), #4                                          ; 3333
                                        0B  13 000DD              BEQL    15$                                                     ;
                          7E    00G 8F  9A 000DF              MOVZBL  #BAS$K_NOTIMP, -(SP)                                    ;
          00000000G       00        01  FB 000E3              CALLS   #1, BAS$$STOP                                           ;
                          64    0C  AE  90 000EA  15$:        MOVB    VALUE, (VALUE_LOCATION)                                 ; 3338
                          5E        10  C0 000EE  16$:        ADDL2   #16, SP                                                 ; 3341
                                        05 000F1              RSB                                                             ;

; Routine Size:  242 bytes,    Routine Base:  _BAS$CODE + 1585
```

```
;  3190          3342  1 GLOBAL ROUTINE BAS$STO_FA_G_R8 (                      ! Store a g floating value
;  3191          3343  1         VALUE0,                                       ! The value to store
;  3192          3344  1         VALUE1,
;  3193          3345  1         DESCRIP : REF BLOCK [8, BYTE],                ! The descriptor to store into
;  3194          3346  1         INDEX1,                                       ! First index
;  3195          3347  1         INDEX2                                        ! Second index
;  3196          3348  1     ) : VA_JSB NOVALUE =
;  3197          3349  1
;  3198          3350  1 !++
;  3199          3351  1 ! FUNCTIONAL DESCRIPTION:
;  3200          3352  1 !
;  3201          3353  1 !       Store a 64-bit g floating value in an array or virtual array.
;  3202          3354  1 !
;  3203          3355  1 ! FORMAL PARAMETERS:
;  3204          3356  1 !
;  3205          3357  1 !       VALUE.rg.v      The value to store
;  3206          3358  1 !                       (Passed as two longwords: VALUE0 and VALUE1)
;  3207          3359  1 !       DESCRIP.rd.da   The descriptor of the array or virtual array
;  3208          3360  1 !       INDEX1.rl.v     The first index into the array
;  3209          3361  1 !       INDEX2.rl.v     The second index into the array
;  3210          3362  1 !
;  3211          3363  1 ! IMPLICIT INPUTS:
;  3212          3364  1 !
;  3213          3365  1 !       NONE
;  3214          3366  1 !
;  3215          3367  1 ! IMPLICIT OUTPUTS:
;  3216          3368  1 !
;  3217          3369  1 !       NONE
;  3218          3370  1 !
;  3219          3371  1 ! ROUTINE VALUE:
;  3220          3372  1 ! COMPLETION CODES:
;  3221          3373  1 !
;  3222          3374  1 !       NONE
;  3223          3375  1 !
;  3224          3376  1 ! SIDE EFFECTS:
;  3225          3377  1 !
;  3226          3378  1 !       Signals if an error is encountered.
;  3227          3379  1 !
;  3228          3380  1 !--
;  3229          3381  1
;  3230          3382  2     BEGIN
;  3231          3383  2
;  3232          3384  2     LOCAL
;  3233          3385  2         BOUNDS : REF VECTOR,
;  3234          3386  2         MULTIPLIERS : REF VECTOR,
;  3235          3387  2         LOW_INDEX,
;  3236          3388  2         HIGH_INDEX,
;  3237          3389  2         INDEX_INCR,
;  3238          3390  2         VALUE_LOCATION,
;  3239          3391  2         INDEX_VALUE,
;  3240          3392  2         INDEX_NUMBER,
;  3241          3393  2         VALUE : VECTOR [2];
;  3242          3394  2
;  3243          3395  2 !+
;  3244          3396  2 ! Put the g floating input value into a local where it will be
;  3245          3397  2 ! safe.
;  3246          3398  2 !-
```

```
;  3247          3399  2     VALUE [0] = .VALUE0;
;  3248          3400  2     VALUE [1] = .VALUE1;
;  3249          3401  !+
;  3250          3402  ! Be sure the array has at least one but no more than two dimensions.
;  3251          3403  !-
;  3252          3404
;  3253          3405        IF ((.DESCRIP [DSC$B_DIMCT] LSSU 1) OR (.DESCRIP [DSC$B_DIMCT] GTRU 2)) THEN BAS$$STOP (BAS$K_ONEOR_TWO)
;  3254          3406
;  3255          3407  !+
;  3256          3408  ! Be sure this array or virtual array holds g floating values.
;  3257          3409  !-
;  3258          3410
;  3259          3411        IF (.DESCRIP [DSC$B_DTYPE] NEQU DSC$K_DTYPE_G) THEN BAS$$STOP (BAS$K_ARGDONMAT);
;  3260          3412
;  3261          3413  !+
;  3262          3414  ! The coefficients and bounds must be present
;  3263          3415  !-
;  3264          3416
;  3265          3417        IF ( NOT (.DESCRIP [DSC$V_FL_COEFF] AND      !
;  3266          3418             .DESCRIP [DSC$V_FL_BOUNDS]))
;  3267          3419        THEN
;  3268          3420            BAS$$STOP (BAS$K_ARGDONMAT);
;  3269          3421
;  3270          3422        MULTIPLIERS = DESCRIP [DSC$L_M1];
;  3271          3423        BOUNDS = DESCRIP [DSC$L_M1] + (%UPVAL*.DESCRIP [DSC$B_DIMCT]);
;  3272          3424  !+
;  3273          3425  ! Compute the lower and upper index numbers based on how the array
;  3274          3426  ! is stored.
;  3275          3427  !-
;  3276          3428
;  3277          3429        IF (.DESCRIP [DSC$V_FL_COLUMN])
;  3278          3430        THEN
;  3279          3431            BEGIN
;  3280          3432            LOW_INDEX = .DESCRIP [DSC$B_DIMCT];
;  3281          3433            HIGH_INDEX = 1;
;  3282          3434            INDEX_INCR = -1;
;  3283          3435            END
;  3284          3436        ELSE
;  3285          3437            BEGIN
;  3286          3438            LOW_INDEX = 1;
;  3287          3439            HIGH_INDEX = .DESCRIP [DSC$B_DIMCT];
;  3288          3440            INDEX_INCR = 1;
;  3289          3441            END;
;  3290          3442
;  3291          3443        INDEX_NUMBER = .LOW_INDEX - .INDEX_INCR;
;  3292          3444  !+
;  3293          3445  ! Compute the linear index from the indices provided.
;  3294          3446  !-
;  3295          3447        VALUE_LOCATION = 0;
;  3296          3448
;  3297          3449        WHILE ((INDEX_NUMBER = .INDEX_NUMBER + .INDEX_INCR) NEQ (.HIGH_INDEX + .INDEX_INCR)) DO
;  3298          3450            BEGIN
;  3299          3451            INDEX_VALUE = (IF (.INDEX_NUMBER EQL 1) THEN .INDEX1 ELSE .INDEX2);
;  3300          3452
;  3301          3453            IF ((.INDEX_VALUE LSS .BOUNDS [(.INDEX_NUMBER - 1)*2])  !
;  3302          3454                OR (.INDEX_VALUE GTR .BOUNDS [((.INDEX_NUMBER - 1)*2) + 1]))
;  3303          3455            THEN
```

```
;  3304          3456  3              BAS$$STOP (BAS$K_SUBOUTRAN);
;  3305          3457  3
;  3306          3458  3          VALUE_LOCATION = (.VALUE_LOCATION+.MULTIPLIERS [.INDEX_NUMBER - 1]) + .INDEX_VALUE;
;  3307          3459  3          END;
;  3308          3460  3
;  3309          3461  3      VALUE_LOCATION = (.VALUE_LOCATION+.DESCRIP [DSC$W_LENGTH]) + .DESCRIP [DSC$A_A0];
;  3310          3462  3  !+
;  3311          3463  3  ! Special handling for virtual arrays.
;  3312          3464  3  !-
;  3313          3465  3
;  3314          3466  3      IF (.DESCRIP [DSC$B_CLASS] EQLU DSC$K_CLASS_BFA)
;  3315          3467  3      THEN
;  3316          3468  3          BEGIN
;  3317          3469  3          BAS$$VA_STORE (.DESCRIP, .VALUE_LOCATION, VALUE);
;  3318          3470  3          END
;  3319          3471  3      ELSE
;  3320          3472  3          BEGIN
;  3321          3473  3
;  3322          3474  3          IF (.DESCRIP [DSC$B_CLASS] NEQU DSC$K_CLASS_A) THEN BAS$$STOP (BAS$K_NOTIMP);
;  3323          3475  3
;  3324          3476  3  !+
;  3325          3477  3  ! Store the value provided into the array
;  3326          3478  3  !-
;  3327          3479  3          BAS$$COPY_G_R1 (VALUE [0], .VALUE_LOCATION);
;  3328          3480  3          END;
;  3329          3481  3
;  3330          3482  1      END;                                        ! end of BAS$STO_FA_G_R8


                5E              14  C2 00000 BAS$STO_FA_G_RB::
                                              SUBL2   #20, SP                         ;  3342
            0C  AE              50  7D 00003          MOVQ    VALUE0, VALUE           ;  3399
                55          0B  A2  9A 00007          MOVZBL  11(DESCRIP), R5         ;  3405
                                05  13 0000B          BEQL    1$
                02              55  91 0000D          CMPB    R5, #2
                                0B  1B 00010          BLEQU   2$
                7E          00G 8F  9A 00012 1$:      MOVZBL  #BAS$K_ONEOR_TWO, -(SP)
    00000000G   00              01  FB 00016          CALLS   #1, BAS$$STOP
                1B          02  A2  91 0001D 2$:      CMPB    2(DESCRIP), #27         ;  3411
                                0B  13 00021          BEQL    3$
                7E          00G 8F  9A 00023          MOVZBL  #BAS$K_ARGDONMAT, -(SP)
    00000000G   00              01  FB 00027          CALLS   #1, BAS$$STOP
05          0A  A2              06  E1 0002E 3$:      BBC     #6, 10(DESCRIP), 4$     ;  3417
                            0A  A2  95 00033          TSTB    10(DESCRIP)             ;  3418
                                0B  19 00036          BLSS    5$
                7E          00G 8F  9A 00038 4$:      MOVZBL  #BAS$K_ARGDONMAT, -(SP) ;  3420
    00000000G   00              01  FB 0003C          CALLS   #1, BAS$$STOP
            04  AE          14  A2  9E 00043 5$:      MOVAB   20(R2), MULTIPLIERS     ;  3422
                56          14 A245 DE 00048          MOVAL   20(DESCRIP)[R5], BOUNDS ;  3423
0B          0A  A2              05  E1 0004D          BBC     #5, 10(DESCRIP), 6$     ;  3429
                51                  55  D0 00052      MOVL    R5, LOW_INDEX           ;  3432
                50                  01  D0 00055      MOVL    #1, HIGH_INDEX          ;  3433
                6E                  01  CE 00058      MNEGL   #1, INDEX_INCR          ;  3434
                                    09  11 0005B      BRB     7$                      ;  3429
```

```
                        51            01  D0 0005D 6$:      MOVL    #1, LOW_INDEX                                   ;3438
                        50            55  D0 00060          MOVL    R5, HIGH_INDEX                                  ;3439
                        6E            01  D0 00063          MOVL    #1, INDEX_INCR                                  ;3440
                55      51            6E  C3 00066 7$:      SUBL3   INDEX_INCR, LOW_INDEX, INDEX_NUMBER             ;3443
                                      57  D4 0006A          CLRL    VALUE_LOCATION                                  ;3447
            08  AE      50            6E  C1 0006C          ADDL3   INDEX_INCR, HIGH_INDEX, 8(SP)                   ;3449
                        55            6E  C0 00071 8$:      ADDL2   INDEX_INCR, INDEX_NUMBER
                    08  AE            55  D1 00074          CMPL    INDEX_NUMBER, 8(SP)
                                      3A  13 00078          BEQL    13$
                        01            55  D1 0007A          CMPL    INDEX_NUMBER, #1                                ;3451
                                      05  12 0007D          BNEQ    9$
                        58            53  D0 0007F          MOVL    INDEX1, INDEX_VALUE
                                      03  11 00082          BRB     10$
                        58            54  D0 00084 9$:      MOVL    INDEX2, INDEX_VALUE
                50      55            01  78 00087 10$:     ASHL    #1, INDEX_NUMBER, R0                            ;3453
                    F8 A640           58  D1 0008B          CMPL    INDEX_VALUE, -8(BOUNDS)[R0]
                                      07  19 00090          BLSS    11$
                    FC A640           58  D1 00092          CMPL    INDEX_VALUE, -4(BOUNDS)[R0]                     ;3454
                                      08  15 00097          BLEQ    12$
                        7E      00G   8F  9A 00099 11$:     MOVZBL  #BAS$K_SUBOUTRAN, -(SP)                         ;3456
            00000000G   00            01  FB 0009D          CALLS   #1, BAS$$STOP
                50      04        AE  04  C3 000A4 12$:     SUBL3   #4, MULTIPLIERS, R0                             ;3458
                51          57        6045  C5 000A9       MULL3   (R0)[INDEX_NUMBER], VALUE_LOCATION, R1
                57          51        58  C1 000AE          ADDL3   INDEX_VALUE, R1, VALUE_LOCATION
                                      BD  11 000B2          BRB     8$                                              ;3449
                        51            62  3C 000B4 13$:     MOVZWL  (DESCRIP), R1                                   ;3461
                        51            57  C4 000B7          MULL2   VALUE_LOCATION, R1
                57      51        10  A2  C1 000BA          ADDL3   16(DESCRIP), R1, VALUE_LOCATION
                    BF  8F        03  A2  91 000BF          CMPB    3(DESCRIP), #191                                ;3466
                                      10  12 000C4          BNEQ    14$
                            0C    AE  9F 000C6             PUSHAB  VALUE                                           ;3469
                          0084    8F  BB 000C9             PUSHR   #^M<R2,R7>
                00000000G   00        03  FB 000CD          CALLS   #3, BAS$$VA_STORE
                                      1E  11 000D4          BRB     16$                                             ;3466
                        04        03  A2  91 000D6 14$:     CMPB    3(DESCRIP), #4                                  ;3474
                                      0B  13 000DA          BEQL    15$
                        7E      00G   8F  9A 000DC          MOVZBL  #BAS$K_NOTIMP, -(SP)
                00000000G   00        01  FB 000E0          CALLS   #1, BAS$$STOP
                        50      0C    AE  9E 000E7 15$:     MOVAB   VALUE, R0                                       ;3479
                        51            57  D0 000EB          MOVL    VALUE_LOCATION, R1
                          00000000G   00  16 000EE          JSB     BAS$$COPY_G_R1
                        5E            14  C0 000F4 16$:     ADDL2   #20, SP                                         ;3482
                                      05  000F7             RSB

; Routine Size:  248 bytes,    Routine Base:  _BAS$CODE + 1677
```

```
;  3332          3483  1 GLOBAL ROUTINE BAS$STO_FA_H_R8 (                      ! Store an h floating value
;  3333          3484  1         VALUE0,                                       ! The value to store
;  3334          3485  1         VALUE1,                                       !
;  3335          3486  1         VALUE2,                                       !
;  3336          3487  1         VALUE3,                                       !
;  3337          3488  1         DESCRIP : REF BLOCK [8, BYTE],                ! The descriptor to store into
;  3338          3489  1         INDEX1,                                       ! First index
;  3339          3490  1         INDEX2                                        ! Second index
;  3340          3491  1     ) : VA_JSB NOVALUE =
;  3341          3492  1
;  3342          3493  1 !++
;  3343          3494  1 ! FUNCTIONAL DESCRIPTION:
;  3344          3495  1 !
;  3345          3496  1 !       Store an h floating value in an array or virtual array.
;  3346          3497  1 !
;  3347          3498  1 ! FORMAL PARAMETERS:
;  3348          3499  1 !
;  3349          3500  1 !       VALUE.rl.v      The value to store
;  3350          3501  1 !                       (Passed as four longwords)
;  3351          3502  1 !       DESCRIP.rd.da   The descriptor of the array or virtual array
;  3352          3503  1 !       INDEX1.rl.v     The first index into the array
;  3353          3504  1 !       INDEX2.rl.v     The second index into the array
;  3354          3505  1 !
;  3355          3506  1 ! IMPLICIT INPUTS:
;  3356          3507  1 !
;  3357          3508  1 !       NONE
;  3358          3509  1 !
;  3359          3510  1 ! IMPLICIT OUTPUTS:
;  3360          3511  1 !
;  3361          3512  1 !       NONE
;  3362          3513  1 !
;  3363          3514  1 ! ROUTINE VALUE:
;  3364          3515  1 ! COMPLETION CODES:
;  3365          3516  1 !
;  3366          3517  1 !       NONE
;  3367          3518  1 !
;  3368          3519  1 ! SIDE EFFECTS:
;  3369          3520  1 !
;  3370          3521  1 !       Signals if an error is encountered.
;  3371          3522  1 !
;  3372          3523  1 !--
;  3373          3524  1
;  3374          3525  2     BEGIN
;  3375          3526  2
;  3376          3527  2     LOCAL
;  3377          3528  2         BOUNDS : REF VECTOR,
;  3378          3529  2         MULTIPLIERS : REF VECTOR,
;  3379          3530  2         LOW_INDEX,
;  3380          3531  2         HIGH_INDEX,
;  3381          3532  2         INDEX_INCR,
;  3382          3533  2         VALUE_LOCATION,
;  3383          3534  2         INDEX_VALUE,
;  3384          3535  2         INDEX_NUMBER,
;  3385          3536  2         VALUE : VECTOR [4];
;  3386          3537  2
;  3387          3538  2 !+
;  3388          3539  2 ! Put the h floating input value into a local where it will be
```

```
;  3389          3540  2 ! safe.
;  3390          3541  2 !-
;  3391          3542  2     VALUE [0] = .VALUE0;
;  3392          3543  2     VALUE [1] = .VALUE1;
;  3393          3544  2     VALUE [2] = .VALUE2;
;  3394          3545  2     VALUE [3] = .VALUE3;
;  3395          3546  2 !+
;  3396          3547  2 ! Be sure the array has at least one but no more than two dimensions.
;  3397          3548  2 !-
;  3398          3549  2
;  3399          3550  2     IF ((.DESCRIP [DSC$B_DIMCT] LSSU 1) OR (.DESCRIP [DSC$B_DIMCT] GTRU 2)) THEN BAS$$STOP (BAS$K_ONEOR_TWO)
;  3400          3551  2
;  3401          3552  2 !+
;  3402          3553  2 ! Be sure this array or virtual array holds h floating values.
;  3403          3554  2 !-
;  3404          3555  2
;  3405          3556  2     IF (.DESCRIP [DSC$B_DTYPE] NEQU DSC$K_DTYPE_H) THEN BAS$$STOP (BAS$K_ARGDONMAT);
;  3406          3557  2
;  3407          3558  2 !+
;  3408          3559  2 ! The coefficients and bounds must be present
;  3409          3560  2 !-
;  3410          3561  2
;  3411          3562  4     IF ( NOT (.DESCRIP [DSC$V_FL_COEFF] AND     !
;  3412          3563  3         .DESCRIP [DSC$V_FL_BOUNDS]))
;  3413          3564  2     THEN
;  3414          3565  2         BAS$$STOP (BAS$K_ARGDONMAT);
;  3415          3566  2
;  3416          3567  2     MULTIPLIERS = DESCRIP [DSC$L_M1];
;  3417          3568  2     BOUNDS = DESCRIP [DSC$L_M1] + (%UPVAL*.DESCRIP [DSC$B_DIMCT]);
;  3418          3569  2 !+
;  3419          3570  2 ! Compute the lower and upper index numbers based on how the array
;  3420          3571  2 ! is stored.
;  3421          3572  2 !-
;  3422          3573  2
;  3423          3574  2     IF (.DESCRIP [DSC$V_FL_COLUMN])
;  3424          3575  2     THEN
;  3425          3576  3         BEGIN
;  3426          3577  3         LOW_INDEX = .DESCRIP [DSC$B_DIMCT];
;  3427          3578  3         HIGH_INDEX = 1;
;  3428          3579  3         INDEX_INCR = -1;
;  3429          3580  3         END
;  3430          3581  2     ELSE
;  3431          3582  3         BEGIN
;  3432          3583  3         LOW_INDEX = 1;
;  3433          3584  3         HIGH_INDEX = .DESCRIP [DSC$B_DIMCT];
;  3434          3585  3         INDEX_INCR = 1;
;  3435          3586  3         END;
;  3436          3587  2
;  3437          3588  2     INDEX_NUMBER = .LOW_INDEX - .INDEX_INCR;
;  3438          3589  2 !+
;  3439          3590  2 ! Compute the linear index from the indices provided.
;  3440          3591  2 !-
;  3441          3592  2     VALUE_LOCATION = 0;
;  3442          3593  2
;  3443          3594  3     WHILE ((INDEX_NUMBER = .INDEX_NUMBER + .INDEX_INCR) NEQ (.HIGH_INDEX + .INDEX_INCR)) DO
;  3444          3595  3         BEGIN
;  3445          3596  3         INDEX_VALUE = (IF (.INDEX_NUMBER EQL 1) THEN .INDEX1 ELSE .INDEX2);
```

```
            IF ((.INDEX_VALUE LSS .BOUNDS [(.INDEX_NUMBER - 1)*2])      !
                OR (.INDEX_VALUE GTR .BOUNDS [((.INDEX_NUMBER - 1)*2) + 1]))
            THEN
                BAS$$STOP (BAS$K_SUBOUTRAN);

            VALUE_LOCATION = (.VALUE_LOCATION*.MULTIPLIERS [.INDEX_NUMBER - 1]) + .INDEX_VALUE;
            END;

        VALUE_LOCATION = (.VALUE_LOCATION*.DESCRIP [DSC$W_LENGTH]) + .DESCRIP [DSC$A_A0];
!+
! Special handling for virtual arrays.
!-

        IF (.DESCRIP [DSC$B_CLASS] EQLU DSC$K_CLASS_BFA)
        THEN
            BEGIN
            BAS$$VA_STORE (.DESCRIP, .VALUE_LOCATION, VALUE);
            END
        ELSE
            BEGIN

            IF (.DESCRIP [DSC$B_CLASS] NEQU DSC$K_CLASS_A) THEN BAS$$STOP (BAS$K_NOTIMP);

!+
! Store the value provided into the array
!-
            BAS$$COPY_H_R3 (VALUE [0], .VALUE_LOCATION);
            END;

        END;                                            ! end of BAS$STO_FA_H_R8
```

```
                5E                 1C  C2 00000 BAS$STO_FA_H_R8::
                                                     SUBL2   #28, SP                          ; 3483
          0C    AE                 50  7D 00003      MOVQ    VALUE0, VALUE                    ; 3542
          14    AE                 52  7D 00007      MOVQ    VALUE2, VALUE+8                  ; 3544
                52        0B       A4  9A 0000B      MOVZBL  11(DESCRIP), R2                  ; 3550
                                   05  13 0000F      BEQL    1$                               ;
                02                 52  91 00011      CMPB    R2, #2                           ;
                                   0B  1B 00014      BLEQU   2$                               ;
                7E        00G      8F  9A 00016 1$:  MOVZBL  #BAS$K_ONEOR_TWO, -(SP)          ;
   00000000G    00                 01  FB 0001A      CALLS   #1, BAS$$STOP                    ;
                1C        02       A4  91 00021 2$:  CMPB    2(DESCRIP), #28                  ; 3556
                                   08  13 00025      BEQL    3$                               ;
                7E        00G      8F  9A 00027      MOVZBL  #BAS$K_ARGDONMAT, -(SP)          ;
   00000000G    00                 01  FB 0002B      CALLS   #1, BAS$$STOP                    ;
05        0A    A4                 06  E1 00032 3$:  BBC     #6, 10(DESCRIP), 4$              ; 3562
                          0A       A4  95 00037      TSTB    10(DESCRIP)                      ; 3563
                                   08  19 0003A      BLSS    5$                               ;
                7E        00G      8F  9A 0003C 4$:  MOVZBL  #BAS$K_ARGDONMAT, -(SP)          ; 3565
   00000000G    00                 01  FB 00040      CALLS   #1, BAS$$STOP                    ;
          04    AE        14       A4  9E 00047 5$:  MOVAB   20(R4), MULTIPLIERS              ; 3567
                53        14 A442      DE 0004C      MOVAL   20(DESCRIP)[R2], BOUNDS          ; 3568
```

```
                    0B      0A   A4              05  E1 00051          BBC     #5, 10(DESCRIP), 6$                     ; 3574
                                 51              52  D0 00056          MOVL    R2, LOW_INDEX                           ; 3577
                                 50              01  D0 00059          MOVL    #1, HIGH_INDEX                          ; 3578
                                 6E              01  CE 0005C          MNEGL   #1, INDEX_INCR                          ; 3579
                                                 09  11 0005F          BRB     7$                                      ; 3574
                                 51              01  D0 00061  6$:     MOVL    #1, LOW_INDEX                           ; 3583
                                 50              52  D0 00064          MOVL    R2, HIGH_INDEX                          ; 3584
                                 6E              01  D0 00067          MOVL    #1, INDEX_INCR                          ; 3585
                        52       51              6F  C3 0006A  7$:     SUBL3   INDEX_INCR, LOW_INDEX, INDEX_NUMBER     ; 3588
                                                 57  D4 0006E          CLRL    VALUE_LOCATION                          ; 3592
                08      AE       50              6E  C1 00070          ADDL3   INDEX_INCR, HIGH_INDEX, 8(SP)           ; 3594
                                 52              6E  C0 00075  8$:     ADDL2   INDEX_INCR, INDEX_NUMBER
                            08   AE              52  D1 00078          CMPL    INDEX_NUMBER, 8(SP)
                                                 3A  13 0007C          BEQL    13$
                                 01              52  D1 0007E          CMPL    INDEX_NUMBER, #1                        ; 3596
                                                 05  12 00081          BNEQ    9$
                                 58              55  D0 00083          MOVL    INDEX1, INDEX_VALUE
                                                 03  11 00086          BRB     10$
                                 58              56  D0 00088  9$:     MOVL    INDEX2, INDEX_VALUE
                        50       52              01  78 0008B  10$:     ASHL    #1, INDEX_NUMBER, R0                    ; 3598
                            F8   A340            58  D1 0008F          CMPL    INDEX_VALUE, -8(BOUNDS)[R0]
                                                 07  19 00094          BLSS    11$
                            FC   A340            58  D1 00096          CMPL    INDEX_VALUE, -4(BOUNDS)[R0]             ; 3599
                                                 0B  15 0009B          BLEQ    12$
                                 7E          00G 8F  9A 0009D  11$:     MOVZBL  #BAS$K_SUBOUTRAN, -(SP)                 ; 3601
                    00000000G    00              01  FB 000A1          CALLS   #1, BAS$$STOP
                        50           04      AE  04  C3 000A8  12$:     SUBL3   #4, MULTIPLIERS, R0                     ; 3603
                        51                   57  6042 C5 000AD          MULL3   (R0)[INDEX_NUMBER], VALUE_LOCATION, R1
                        57                   51  58  C1 000B2          ADDL3   INDEX_VALUE, R1, VALUE_LOCATION
                                                 BD  11 000B6          BRB     8$                                      ; 3594
                                 51              64  3C 000B8  13$:     MOVZWL  (DESCRIP), R1                           ; 3606
                                 51              57  C4 000BB          MULL2   VALUE_LOCATION, R1
                        57       51          10  A4  C1 000BE          ADDL3   16(DESCRIP), R1, VALUE_LOCATION
                            BF   8F          03  A4  91 000C3          CMPB    3(DESCRIP), #191                        ; 3611
                                                 10  12 000C8          BNEQ    14$
                                             0C  AE  9F 000CA          PUSHAB  VALUE                                   ; 3614
                                          0090   8F  BB 000CD          PUSHR   #^M<R4,R7>
                        00000000G        00          03  FB 000D1          CALLS   #3, BAS$$VA_STORE
                                                 1E  11 000D8          BRB     16$                                     ; 3611
                                 04          03  A4  91 000DA  14$:     CMPB    3(DESCRIP), #4                          ; 3619
                                                 0B  13 000DE          BEQL    15$
                                 7E          00G 8F  9A 000E0          MOVZBL  #BAS$K_NOTIMP, -(SP)
                        00000000G        00          01  FB 000E4          CALLS   #1, BAS$$STOP
                                 50          0C  AE  9E 000EB  15$:     MOVAB   VALUE, R0                               ; 3624
                                 51              57  D0 000EF          MOVL    VALUE_LOCATION, R1
                                     00000000G   00  16 000F2          JSB     BAS$$COPY_H_R3
                                 5E              1C  C0 000F8  16$:     ADDL2   #28, SP                                 ; 3627
                                                 05  000FB              RSB

; Routine Size:  252 bytes,    Routine Base:  _BAS$CODE + 176F

;  3477          3628  1
;  3478          3629  1 END                                     ! end of module BAS$VIRTUAL_ARR
;  3479          3630  1
;  3480          3631  0 ELUDOM
```

```
;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  _BAS$CODE                      6251  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)


;                                Library Statistics
;
;                                          --------- Symbols ---------       Pages       Processing
;       File                               Total     Loaded    Percent       Mapped      Time
;
;  _$255$DUA28:[SYSLIB]STARLET.L32;1        9776         26          0        581          00:01.0



;                                   COMMAND QUALIFIERS
;
;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:BASVIRTUA/OBJ=OBJ$:BASVIRTUA MSRC$:BASVIRTUA/UPDATE=(ENH$:BASVIRTUA
;       )

; Size:          6251 code + 0 data bytes
; Run Time:          01:53.8
; Elapsed Time:      03:55.5
; Lines/CPU Min:      1915
; Lexemes/CPU-Min: 17555
; Memory Used:  245 pages
; Compilation Complete
```

BASUVRTUA
LIS

BASUDFWL
LIS

BASUNLOCK
LIS

BASUVECTOR
LIS

BASUAL
LIS

BASUIRTIO
LIS

BASUNWIND BASUPDATE
LIS

BASUEXTR2
LIS

BASXLATE
LIS

BPAMESAG
LIS

BPAF3S
LIS

BPAGE1BLK
LIS

BASZ1RE
LIS